



A memetic procedure for global multi-objective optimization

Matteo Lapucci¹ · Pierluigi Mansueto¹ · Fabio Schoen¹

Received: 27 January 2022 / Accepted: 2 November 2022
© The Author(s) 2022

Abstract

In this paper we consider multi-objective optimization problems over a box. Several computational approaches to solve these problems have been proposed in the literature, that broadly fall into two main classes: evolutionary methods, which are usually very good at exploring the feasible region and retrieving good solutions even in the nonconvex case, and descent methods, which excel in efficiently approximating good quality solutions. In this paper, first we confirm, through numerical experiments, the advantages and disadvantages of these approaches. Then we propose a new method which combines the good features of both. The resulting algorithm, which we call Non-dominated Sorting Memetic Algorithm, besides enjoying interesting theoretical properties, excels in all of the numerical tests we performed on several, widely employed, test functions.

Keywords Multi-objective optimization · Memetic algorithm · NSGA-II · Descent method · Pareto front approximation

Mathematics Subject Classification 90C29 · 90C30 · 68W20

1 Introduction

Multi-objective optimization problems have a significant relevance in many applications of various fields, such as engineering [5, 47, 50], management [28, 55], statistics

✉ Pierluigi Mansueto
pierluigi.mansueto@unifi.it

Matteo Lapucci
matteo.lapucci@unifi.it

Fabio Schoen
fabio.schoen@unifi.it

¹ Global Optimization Laboratory (GOL), Department of Information Engineering, University of Florence, Via di Santa Marta, 3, 50139 Florence, Italy

[7], space exploration [45, 51], etc.. It is thus not surprising that many research streams have flourished around this topic for the last 25 years.

Multi-objective optimization (MOO) exhibits two major complexities, that coupled together make problems particularly difficult to handle. The first complexity element is the general absence of a solution minimizing all the objective functions simultaneously; as a consequence, the definitions of optimality (global, local and stationarity), based on Pareto's theory, are not trivial and make optimization processes not obvious, both in terms of aims and tools. The second one, on the other hand, traces one classical issue typical of scalar optimization: in absence of convexity assumptions, there is no equivalence between local and global (Pareto) optimality.

The combination of the two aforementioned features of MOO problems makes evolutionary-type algorithms (EAs) particularly well suited to be used and, indeed, they have been the most widely studied class of algorithms for decades in this context [35, 44]. The NSGA-II algorithm [13] is arguably the most popular among these methods; basically, it is a population-based procedure exploiting a cheaply computable score to efficiently rank solutions w.r.t. the objectives, and performing the classical genetic crossover, mutation and selection operations to create the new generation of solutions. Basically, NSGA-II represents the de facto standard, at least popularity-wise, for unconstrained and bound-constrained MOO.

In fact, alongside the EA stream, two different classes of approaches have been studied for MOO. The first one concerns scalarization approaches [17, 18, 46]. However, solving MOO problems by scalarization has some drawbacks: firstly, unfortunate choices of weights may lead to unbounded scalar problems, even under strong regularity assumptions [20, Section 7]; moreover, scalarization is designed to produce a single solution and, in order to generate an approximation of the whole Pareto front, the problem has to be repeatedly solved with different choices of weights; unfortunately, it is not known a priori how weights should be selected to obtain a wide and uniform Pareto front.

The other family of methods is that of MO descent methods (either first-order, second-order and derivative-free) [6, 8, 16, 20, 21, 26, 27]. These methods mimic classical iterative scalar optimization algorithms.

Originally, these methods were designed to generate a single Pareto-stationary solution, but in recent years specific strategies have been proposed to handle lists of points and to generate approximations of the entire Pareto front [9–12, 22, 38]. Numerical results show that these methods, when used on problems with reasonable regularity assumptions, are effective and much more efficient than evolutionary methods, especially as the problem size grows [9, 10, 12].

On the other hand, descent algorithms have convergence properties that are theoretically relevant but, in practice, guarantee only Pareto stationarity of the retrieved solutions. This is a significant limitation with highly non-convex problems, similarly as for gradient-based algorithms in scalar optimization that may converge to stationary points which are not even local optima.

In scalar global optimization, particularly successful strategies are memetic ones. Memetic algorithms combine population-based techniques (either heuristic and/or genetic ones) and local search steps [4, 29, 30, 39, 40, 43]. In the case of MOO, this idea has only superficially been considered. In fact, each class of MOO algorithms

has practical drawbacks. EAs have no theoretical convergence property [20] and are usually expensive [31, 37, 49]. On the other hand, descent algorithms often produce suboptimal solutions when starting from non carefully chosen points and are thus not suitable for highly non-convex problems. For these reasons, some MO memetic approaches have been proposed in the literature. However, we can find approaches that are mostly application-specific [54] or that employ heuristic [33, 34, 36, 41], meta-heuristic [1, 53], stochastic [15, 19] or scalarization-based [31, 49, 52] local search steps. Even the few proposed strategies employing gradient information for the local search steps do not exploit the concept of common descent directions. Rather, convex combinations of gradients are generated and exploited in various ways [2, 3, 34, 37, 48].

In this paper, we show by computational experiments the benefits and limitations of evolutionary and MO descent algorithms in different settings (high/low dimensionality, convex/non-convex objectives). Then, we propose a memetic algorithm for bound-constrained MOO problems, combining an evolutionary approach (namely, the popular NSGA-II) with MO descent methods, similarly to what is done in the scalar case in [30]. We finally show that the proposed method, that inherits the good features of both EA and MO descent families, outperforms the state-of-the-art MOO solvers in any setting.

The rest of the manuscript is organized as follows. In Sect. 2, we recall the main concepts of both the descent methods and the NSGA-II algorithm. In Sect. 3, we provide a description of our memetic algorithm along with a theoretical analysis. In Sect. 4, we first compare the two families of algorithms in some specific problems, in order to show the benefits and the shortcomings of both. Then, we show the results of computational experiments highlighting the good performance of our approach w.r.t. main state-of-the-art methods. Finally, in Sect. 5 we provide some concluding remarks.

2 Preliminaries

In this work, we consider multi-objective optimization problems of the form

$$\begin{aligned} \min_{x \in \mathbb{R}^n} F(x) &= (f_1(x), \dots, f_m(x))^T \\ \text{s.t. } x &\in [l, u], \end{aligned} \tag{1}$$

where $F : \mathbb{R}^n \rightarrow \mathbb{R}^m$ is a continuously differentiable function and $l, u \in \mathbb{R}^n$ with $l_i \leq u_i \forall i \in \{1, \dots, n\}$. The values of l and u may possibly be infinite. Given the boundary constraints, we denote the feasible set, which is closed, convex and non empty, by $\Omega = \{x \in \mathbb{R}^n \mid x \in [l, u]\}$. We denote by J_F the Jacobian matrix associated with F .

In order to introduce some preliminary concepts of multi-objective optimization, we define a partial ordering of the points in \mathbb{R}^m . Considering two points $u, v \in \mathbb{R}^m$, we define

$$\begin{aligned} u < v &\iff u_i < v_i & \forall i = 1, \dots, m, \\ u \leq v &\iff u_i \leq v_i & \forall i = 1, \dots, m. \end{aligned}$$

If $u \leq v$ and $u \neq v$, we can say that u dominates v and we use the following notation: $u \preceq v$. Finally, we say that the point $x \in \mathbb{R}^n$ dominates $y \in \mathbb{R}^n$ w.r.t. F if $F(x) \preceq F(y)$.

In multi-objective optimization problems, we ideally would like to obtain a point which simultaneously minimizes all the objectives f_1, \dots, f_m . However, such a solution is unlikely to exist. For this reason, the Pareto optimality concepts have been introduced.

Definition 1 A point $\bar{x} \in \Omega$ is *Pareto optimal* for Problem (1) if a point $y \in \Omega$ such that $F(y) \preceq F(\bar{x})$ does not exist. If there exists a neighborhood $\mathcal{N}(\bar{x})$ such that the previous property is satisfied in $\Omega \cap \mathcal{N}(\bar{x})$, then \bar{x} is *locally Pareto optimal*.

In practice, it is difficult to attain solutions characterized by the Pareto optimality property. A slightly weaker property is weak Pareto optimality.

Definition 2 A point $\bar{x} \in \Omega$ is *weakly Pareto optimal* for Problem(1) if a point $y \in \Omega$ such that $F(y) < F(\bar{x})$ does not exist. If there exists a neighborhood $\mathcal{N}(\bar{x})$ such that the previous property is satisfied in $\Omega \cap \mathcal{N}(\bar{x})$, then \bar{x} is *locally weakly Pareto optimal*.

We refer to the set of all Pareto optimal solutions of the problem as the *Pareto set*, while by *Pareto front* we refer to the image of the Pareto set through F .

We can now introduce the concept of Pareto stationarity.

Definition 3 A point $\bar{x} \in \Omega$ is *Pareto-stationary* for Problem(1) if we have that

$$\max_{j=1, \dots, m} \nabla f_j(\bar{x})^T d \geq 0,$$

for all feasible directions $d \in \mathcal{D}(\bar{x}) = \{v \in \mathbb{R}^n \mid \exists \bar{t} > 0 : \bar{x} + tv \in \Omega \forall t \in [0, \bar{t}]\}$.

Under differentiability assumptions, Pareto stationarity is a necessary condition for all types of Pareto optimality. Further assuming the convexity of the objectives in Problem (1), the condition is also sufficient for Pareto optimality. The property can be compactly re-written as

$$\min_{d \in \mathcal{D}(\bar{x})} \max_{j=1, \dots, m} \nabla f_j(\bar{x})^T d = 0.$$

Finally, we introduce a relaxation of Pareto stationarity, recalling the ε -Pareto-stationarity concept introduced in [8].

Definition 4 Let $\varepsilon \geq 0$. A point $\bar{x} \in \Omega$ is ε -Pareto-stationary for Problem(1) if

$$\min_{\substack{d \in \mathcal{D}(\bar{x}) \\ \|d\| \leq 1}} \max_{j=1, \dots, m} \nabla f_j(\bar{x})^T d \geq -\varepsilon.$$

In the following, we briefly review evolutionary and descent algorithms for MOO, with particular emphasis on the NSGA-II algorithm and steepest/projected gradient descent methods respectively.

2.1 Multi-objective descent methods

First of all, let us address the following unconstrained optimization problem.

$$\min_{x \in \mathbb{R}^n} F(x) = (f_1(x), \dots, f_m(x))^T. \tag{2}$$

If a point $\bar{x} \in \mathbb{R}^n$ is not Pareto-stationary, then there exists a descent direction w.r.t. all the objectives. Therefore, following [21, Section 3.1], we can introduce the *steepest common descent direction* as the solution of problem

$$\min_{\substack{d \in \mathbb{R}^n \\ \|d\| \leq 1}} \max_{j=1, \dots, m} \nabla f_j(\bar{x})^T d, \tag{3}$$

which, if ℓ_∞ norm is employed, can be re-written as an LP one:

$$\begin{aligned} \min_{\beta \in \mathbb{R}, d \in \mathbb{R}^n} \quad & \beta \\ \text{s.t.} \quad & -1 \leq d_i \leq 1 \quad \forall i = 1, \dots, n, \\ & \nabla f_j(\bar{x})^T d \leq \beta \quad \forall j = 1, \dots, m. \end{aligned} \tag{4}$$

A slightly different problem formulation is the ℓ_2 -regularized one, which is again proposed in [21]. However, we preferred to use formulation (3) because of the simplicity of the LP problem. We define the continuous function $\theta : \mathbb{R}^n \rightarrow \mathbb{R}$ such that $\theta(\bar{x})$ indicates the optimal value of Problem (3) at \bar{x} . If \bar{x} is Pareto-stationary, $\theta(\bar{x}) = 0$, otherwise $\theta(\bar{x}) < 0$. We also denote by $\mathcal{V}(\bar{x}) \subseteq \mathbb{R}^n$ the set of optimal solutions to Problem (3). Indeed, the solution may not be unique, although this fact is not a real technical issue.

Based on the concept of steepest common descent direction, the standard *Multi-Objective Steepest Descent* (MOSD) algorithm was proposed in [21]. In MOSD, a backtracking *Armijo-type Line Search* (ALS) is used. The idea of this latter one is to reduce the step size until we get a sufficient decrease for all the objective functions. We report ALS in Algorithm 1.

We now recall the main theoretical results of the two algorithms, starting from the finite termination property of the line search.

Lemma 1 [21, Lemma 4] *If F is continuously differentiable and $J_F(x)d < 0$ (i.e. $\theta(x) < 0$), then there exists some $\varepsilon > 0$, which may depend on x, d and β , such that*

$$F(x + td) < F(x) + \beta t J_F(x)d$$

for all $t \in (0, \varepsilon]$.

Regarding the MOSD procedure, the following convergence property holds.

Lemma 2 [21, Theorem 1, Section 9.1] *Every accumulation point of the sequence $\{x_k\}$ produced by the MOSD algorithm is a Pareto-stationary point. If the function F has bounded level sets, in the sense that $\{x \in \mathbb{R}^n \mid F(x) \leq F(x_0)\}$ is bounded, then the sequence $\{x_k\}$ stays bounded and has at least one accumulation point.*

Algorithm 1: Armijo-type Line Search

```

1 Input:  $F : \mathbb{R}^n \rightarrow \mathbb{R}^m$ ,  $x_k \in \mathbb{R}^n$ ,  $d_k \in \mathcal{V}(x_k)$ ,  $\alpha_0 > 0$ ,  $\delta \in (0, 1)$ ,  $\beta \in (0, 1)$ .
2  $\alpha = \alpha_0$ 
3 while  $F(x_k + \alpha d_k) \not\leq F(x_k) + \beta \alpha J_F(x_k) d_k$  do
4   |  $\alpha = \delta \alpha$ 
5 return  $\alpha$ 

```

Through the years, the MOSD procedure was extended to handle a sequence of sets $\{X_k\}$ of non-dominated points, rather than a sequence of points, aiming to approximate the Pareto front of the optimization problems. Indeed, in the multi-objective context, an approximation of the Pareto front could be much more useful than a single solution: the user is free to choose, a posteriori, the solution providing the most appropriate trade-off among many. An algorithm representing an extension of the MOSD procedure is the one introduced in [10], which is called *Front Steepest Descent Algorithm* (FSDA).

In the next lemma, we report the convergence property of the algorithm, where the authors use the concept of *linked sequence* introduced in [38].

Definition 5 A sequence $\{x_k\}$ is a linked sequence if, for all k , $x_k \in X_k$ and x_k is generated at iteration $k - 1$ starting the search procedure from x_{k-1} .

Lemma 3 [10, Proposition 5] *Let $\{X_k\}$ be the sequence of sets of non-dominated points produced by FSDA. Let us assume that there exists a point $x_0 \in X_0$ such that:*

- x_0 is not Pareto-stationary;
- the set $\mathcal{L}(x_0) = \bigcup_{j=1}^m \{x \in \mathbb{R}^n : f_j(x) \leq f_j(x_0)\}$ is compact.

Let $\{x_k\}$ be a linked sequence, then it admits limit points and every limit point is Pareto-stationary for Problem (2).

In [10], two concepts are introduced. The first one is the *steepest partial descent direction* at \bar{x} w.r.t. a subset of indices of objectives $I \subseteq \{1, \dots, m\}$. This type of direction at a point \bar{x} can be found solving this optimization problem:

$$\min_{\substack{d \in \mathbb{R}^n \\ \|d\| \leq 1}} \max_{j \in I} \nabla f_j(\bar{x})^T d. \quad (5)$$

As for the steepest common descent direction, we define the continuous function $\theta^I : \mathbb{R}^n \rightarrow \mathbb{R}$, where $\theta^I(\bar{x})$ indicates the optimal value of Problem (5) at \bar{x} . Accordingly, we denote by $\mathcal{V}^I(\bar{x}) \subseteq \mathbb{R}^n$ the set of optimal solutions to the Problem(5). If appropriately used, the steepest partial descent direction may be useful to spread the search in the objectives space and to reach the extreme regions of the Pareto front.

For our purposes, given a subset I , we also define $F_I(\bar{x})$ as the $|I|$ -dimensional vector with components $f_j(\bar{x})$, with $j \in I$. In addition, given a set of points \bar{X} , we introduce the set $\bar{X}^I \subseteq \bar{X}$ as the set of points that are mutually non-dominated w.r.t. F_I , i.e.

$$\bar{X}^I = \{x \in \bar{X} \mid \nexists y \in \bar{X} \text{ s.t. } F_I(y) \leq F_I(x)\}.$$

The second concept introduced in [10] is a weaker front-based variant of ALS: we call it the *Front Armijo-Type Line Search* (FALS). We report it in Algorithm 2. In FALS, the step size is reduced until a sufficient decrease is reached w.r.t. all the points in \bar{X}^I for at least one of the objective functions f_j , with $j \in I$. FALS can be considered as a weak extension of ALS to the multi-objective case. As it is not required to obtain a sufficient decrease for all the objective functions, employing FALS leads to two consequences: less required computational time and bigger values for the step size. These features again may be very useful to obtain good and spread Pareto front approximations in a short time.

Algorithm 2: Front Armijo-type Line Search

```

1 Input:  $F : \mathbb{R}^n \rightarrow \mathbb{R}^m$ ,  $I \subseteq \{1, \dots, m\}$ ,  $X_k^I$  set of mutually non-dominated points w.r.t.  $F_I$ ,  $x_c \in X_k^I$ ,
    $d_c^I \in \mathcal{V}^I(x_c)$ ,  $\theta^I(x_c) \in \mathbb{R}$ ,  $\alpha_0 > 0$ ,  $\delta \in (0, 1)$ ,  $\beta \in (0, 1)$ .
2  $\alpha = \alpha_0$ 
3 while  $\exists y \in X_k^I$  s.t.  $F_I(y) + \mathbf{1}\beta\alpha\theta^I(x_c) < F_I(x_c + \alpha d_c^I)$  do
4   |  $\alpha = \delta\alpha$ 
5 return  $\alpha$ 

```

FALS has a finite termination property, which we recall in the following lemma.

Lemma 4 [10, Proposition 4] *Let $I \subseteq \{1, \dots, m\}$, $x_c \in X_k^I$ be such that $\theta^I(x_c) < 0$, i.e. there exists a direction $d_c^I \in \mathcal{V}^I(x_c)$ such that*

$$\nabla f_j(x_c)^T d_c^I < 0$$

$\forall j \in I$. Then $\exists \bar{\alpha} > 0$, sufficiently small, such that

$$F_I(y) + \mathbf{1}\beta\bar{\alpha}\theta^I(x_c) \not\leq F_I(x_c + \bar{\alpha}d_c^I), \quad \forall y \in X_k^I,$$

i.e., the while loop of FALS terminates in a finite number \bar{h} of iterations, returning a value $\bar{\alpha} = \delta^{\bar{h}}\alpha_0$. Furthermore, the produced point $x_c + \bar{\alpha}d_c^I$ is not dominated with respect to the set X_k .

Since we consider bound constraints in Problem(1), we need to recall the (single-point) *Multi-Objective Projected Gradient* (MOPG) method. This latter algorithm was firstly introduced in [16] and then developed and analyzed in [24, 25]. In addition, the MOPG main results were summarized in [23]. The method is an extension of the MOSD procedure dealing with constrained problems. In particular, it deals with optimization problems characterized by a feasible closed and convex set Ω . We report the MOPG procedure in Algorithm 3.

The first difference w.r.t. the MOSD procedure is the way the direction is retrieved, since now the problem constraints have to be considered. The steepest descent direction is now defined as the solution of

$$\min_{\substack{z \in \Omega \\ \|z - \bar{x}\| \leq 1}} \max_{j=1, \dots, m} \nabla f_j(\bar{x})^T (z - \bar{x}). \tag{6}$$

Algorithm 3: Multi-Objective Projected Gradient

```

1 Input:  $F : \mathbb{R}^n \rightarrow \mathbb{R}^m$ ,  $\Omega$  feasible closed and convex set,  $x_0 \in \Omega$ .
2  $k = 0$ 
3 while  $x_k$  is not Pareto-stationary do
4   Compute
      
$$z_{\Omega k} \in \underset{\substack{z \in \Omega \\ \|z - x_k\| \leq 1}}{\operatorname{argmin}} \max_{j=1, \dots, m} \nabla f_j(x_k)^T (z - x_k)$$

5    $d_{\Omega k} = z_{\Omega k} - x_k$ 
6    $\alpha_k = \text{ALS}(F(\cdot), x_k, d_{\Omega k})$ 
7    $x_{k+1} = x_k + \alpha_k d_{\Omega k}$ 
8    $k = k + 1$ 
9 return  $x_k$ 

```

Note that, again, in practice we employ the ℓ_∞ norm so that the problem can be reformulated as an LP problem similar to (4). As in the unconstrained case, we define the continuous function $\theta_\Omega : \mathbb{R}^n \rightarrow \mathbb{R}$ such that $\theta_\Omega(\bar{x})$ indicates the optimal value of Problem(6) at \bar{x} . We also denote by $\mathcal{Z}_\Omega(\bar{x}) \subseteq \Omega$ the set of optimal solutions to Problem(6), and by $\mathcal{V}_\Omega(\bar{x}) = \{z - \bar{x} \mid z \in \mathcal{Z}_\Omega(\bar{x})\} \subseteq \Omega$ the set of optimal directions. We denote these latter ones as *constrained steepest common descent directions*; if a subset $I \subseteq \{1, \dots, m\}$ is considered, they are referred to *constrained steepest partial descent directions*. If $\theta_\Omega(\bar{x}) < 0$, the point is not Pareto-stationary and we proceed to find a step size through ALS. As opposed to the unconstrained case, where α_0 can be any positive real number, in the MOPG procedure $\alpha_0 = 1$. Since the set Ω is convex, d is a feasible direction by construction and $\alpha_0 = 1$, every produced point will be feasible.

We report here two theoretical results of the MOPG method.

Lemma 5 [23, Lemma 4.3] *Let $\{x_k\} \subset \mathbb{R}^n$ be a sequence generated by MOPG. Then, we have $\{x_k\} \subset \Omega$.*

Lemma 6 [23, Theorem 4.4] *Every accumulation point, if any, of a sequence $\{x_k\}$ generated by MOPG is a feasible Pareto-stationary point.*

In order to deal with box-constrained optimization problems, we adapted the FSDA algorithm [10]. We call the adaptation *Front Projected Gradient Algorithm* (FPGA) and the differences w.r.t. the original algorithm are the following:

- the initial set X_0 is composed by *feasible* non-dominated points w.r.t. F ;
- the direction is found solving Problem(6) instead of Problem(3);
- we employ the *Bound-constrained Front Armijo-Type Line Search* (B-FALS), which is a modified version of FALS that we introduced to take into account bound constraints.

B-FALS, which we report in Algorithm 4, is similar to FALS: the only added requirement is that the step size must lead to a point that is feasible.

Through the following proposition, we show that B-FALS terminates in a finite number of iterations.

Algorithm 4: Bound-constrained Front Armijo-type Line Search

1 Input: $F : \mathbb{R}^n \rightarrow \mathbb{R}^m$, Ω feasible convex set, $I \subseteq \{1, \dots, m\}$, X_k^I set of feasible mutually non-dominated points w.r.t. F_I , $x_c \in X_k^I$, $d_{\Omega_c}^I \in \mathcal{V}_{\Omega}^I(x_c)$, $\theta_{\Omega}^I(x_c) \in \mathbb{R}$, $\alpha_0 > 0$, $\delta \in (0, 1)$, $\beta \in (0, 1)$.
 2 $\alpha = \alpha_0$
 3 **while** $x_c + \alpha d_{\Omega_c}^I \notin \Omega$ **or** $\exists y \in X_k^I$ s.t. $F_I(y) + \mathbf{1}\beta\alpha\theta_{\Omega}^I(x_c) < F_I(x_c + \alpha d_{\Omega_c}^I)$ **do**
 4 $\alpha = \delta\alpha$
 5 **return** α

Proposition 1 Let $I \subseteq \{1, \dots, m\}$, $x_c \in X_k^I$ be such that $\theta_{\Omega}^I(x_c) < 0$, i.e. there exists a direction $d_{\Omega_c}^I \in \mathcal{V}_{\Omega}^I(x_c)$ such that

$$\nabla f_j(x_c)^T d_{\Omega_c}^I < 0$$

$\forall j \in I$. Then $\exists \bar{\alpha} > 0$, sufficiently small, such that

$$x_c + \bar{\alpha} d_{\Omega_c}^I \in \Omega$$

and

$$F_I(y) + \mathbf{1}\beta\bar{\alpha}\theta_{\Omega}^I(x_c) \not\prec F_I(x_c + \bar{\alpha} d_{\Omega_c}^I) \quad \forall y \in X_k^I,$$

i.e., the while loop of B-FALS terminates in a finite number \bar{h} of iterations, returning a value $\bar{\alpha} = \delta^{\bar{h}}\alpha_0$. Furthermore, the produced point $x_c + \bar{\alpha} d_{\Omega_c}^I$ is not dominated with respect to the set X_k .

Proof Assume by contradiction that the thesis is false. Then the algorithm produces an infinite sequence $\{\delta^h\alpha_0\}$ such that, for all h , either

$$x_c + \delta^h\alpha_0 d_{\Omega_c}^I \notin \Omega$$

or a point $y_h \in X_k^I$ exists such that

$$F_I(y_h) + \mathbf{1}\beta\delta^h\alpha_0\theta_{\Omega}^I(x_c) < F_I(x_c + \delta^h\alpha_0 d_{\Omega_c}^I). \tag{7}$$

Since Ω is convex, $d_{\Omega_c}^I$ is a feasible direction by construction and $\delta^h\alpha_0 \rightarrow 0$ as $h \rightarrow \infty$, for h sufficiently large the point $x_c + \delta^h\alpha_0 d_{\Omega_c}^I$ is feasible and thus Condition (7) holds. Then, following the proof of Proposition 4 in [10], we can prove the thesis. \square

We provide a full description of FPGA in Appendix A, along with feasibility and convergence properties. In this work, we consider the FPGA algorithm as a representative of gradient-based methods designed to produce Pareto front approximations for bound-constrained optimization problems.

In [10], a variant of the FSDA algorithm exploiting an Armijo-type extrapolation technique is also introduced. The authors claim that this variant outperforms the original algorithm. For this reason, in the initial stage of our work, we decided to also adapt this variant for box-constrained optimization problems. However, some preliminary computational experiments led us to the somewhat surprising conclusion that vanilla FPGA is better than using extrapolation. A possible justification of this result may lie in the presence of constraints. Anyhow, we decided for the sake of brevity to only consider FPGA in the remainder of the paper.

2.2 NSGA-II

NSGA-II is a non-dominated sorting-based multi-objective evolutionary algorithm that was proposed in [13]. In particular, NSGA-II is a genetic algorithm that creates a mating pool by combining the parent and offspring populations and selecting the best N solutions. In this section, we review the main characteristics of NSGA-II. For a deeper understanding of the algorithm mechanisms, the reader is referred to the original work [13].

We report the main steps of NSGA-II in Algorithm 5.

Algorithm 5: NSGA-II

```

1 Input:  $F : \mathbb{R}^n \rightarrow \mathbb{R}^m$ ,  $\Omega = \{x \in \mathbb{R}^n \mid x \in [l, u]\}$  feasible set,  $X_0 \subset \Omega$ ,  $N$  population size.
2  $k = 0$ 
3  $\hat{X}_0 = X_0$ 
4  $\hat{R}_0, \hat{C}_0 = \text{getMetrics}(\hat{X}_0)$ 
5  $X_0, R_0, C_0 = \text{getSurvivors}(\hat{X}_0, \hat{R}_0, \hat{C}_0, N)$ 
6 while a stopping criterion is not satisfied do
7    $P_k = \text{getParents}(X_k, R_k, C_k)$ 
8    $O_k = \text{crossover}(P_k, l, u)$ 
9    $\tilde{O}_k = \text{mutation}(O_k, l, u)$ 
10   $\hat{X}_{k+1} = X_k \cup \tilde{O}_k$ 
11   $\hat{R}_{k+1}, \hat{C}_{k+1} = \text{getMetrics}(\hat{X}_{k+1})$ 
12   $X_{k+1}, R_{k+1}, C_{k+1} = \text{getSurvivors}(\hat{X}_{k+1}, \hat{R}_{k+1}, \hat{C}_{k+1}, N)$ 
13   $k = k + 1$ 
14 return  $X_k$ 

```

NSGA-II deals with a fixed size population (N solutions) and takes as input an initial population X_0 . For the sake of clarity, from now on we consider X_0 as a set composed by N feasible solutions. However, we want to remark two facts.

- Starting with a population X_0 composed by N points is not necessary: if the population is smaller/bigger, after the first iteration it is increased/reduced in order to get exactly N solutions in it.
- NSGA-II can also manage unfeasible points. However, since in our work we address bound constrained problems, and the genetic operators ensure that after

the first iteration no point in the population violates the bound constraints, we assume that X_0 is only composed by feasible points.

The core idea of the algorithm is that during an iteration:

- the parents are chosen among the current solutions (Line 7);
- N offsprings are created from the parents through the `crossover` operator (Line 8);
- the offsprings are mutated using the `mutation` function (Line 9);
- a new population of $2N$ solutions is created merging the current population with the offsprings (Line 10);
- by the function `getMetrics` scores are associated to the members (Line 11);
- only the best N points (survivors) are selected and maintained (Line 12).

The `crossover` and `mutation` operators have a crucial role in the NSGA-II mechanisms. The `crossover` operator aim is the creation of offsprings that inherit (hopefully the best) features of the parents. The `mutation` operator introduces some random changes in the offsprings. This latter one could be useful when we want to spread our search in the objectives space as much as possible. For a more detailed and technical explanation about these two operators, we again refer the reader to [13]. We want to remark here that the NSGA-II mechanisms ensure that there are no duplicates among the offsprings and any offspring is not a duplicate of any point in the current population. At the end of the algorithm execution, the current population X_k is returned.

In the next subsections, we provide other details of the algorithm that are useful for our purposes.

2.2.1 Metrics

In this section, we explain the metrics used in the NSGA-II mechanisms (computed in the `getMetrics` function). In particular, these scores are used to select the parents and the survivors.

The first one is the *ranking*, which leads to a splitting of the population in different domination levels. Briefly, if a point has rank 0, it means that it is not dominated by any point in X_k w.r.t. F . If it has rank 1, it is dominated by some of the points with rank 0, but it is not w.r.t. any other point with rank equal to or greater than 1. In order to obtain the ranking values, a fast sorting approach is employed, which is one of the strength elements of the NSGA-II algorithm.

The second considered metric is *crowding distance*. It is useful to get an estimate of the density of solutions surrounding a particular point in the population. Having a high crowding distance indicates that the point is in a poorly populated area of the objectives space, and maintaining it in the population may likely lead to a spread Pareto front. Note that for each point this metric is calculated with respect to the solutions with the same ranking value.

We again refer the reader to the original paper [13] for the rigorous definition of the metrics.

2.2.2 Parents selection

In the `getParents` function, pairs of parents are randomly chosen among the solutions in X_k . Then, considering a pair, only one of the two points is selected by *binary tournament*. In this latter one, the solutions are compared in the following way.

- The point with the lowest rank is preferred.
- If the ranking value is the same for both points, the one with the highest crowding distance is chosen.
- In the unlikely case in which the crowding distance values are equal too, a random choice is done.

The selected point will be used with a parent chosen from another pair in the `crossover` function in order to create offsprings.

This approach of comparing the solutions is also used in the `getSurvivors` function.

2.2.3 Selection operation

After getting the offsprings through the `crossover` and `mutation` operators, the new population is composed by $2N$ solutions. The aim of the `getSurvivors` function is to select and maintain the best N solutions. As in the `getParents` function, the selection is based on the ranking and the crowding distance.

- The set composed by the $2N$ points is initially sorted based on the ranking.
- The solutions with the same rank are sorted based on the crowding distance.
- The first N points are chosen as the best ones.

3 Non-dominated sorting memetic algorithm

In this section, we introduce a novel memetic algorithm for bound-constrained MOO problems, which we call *Non-dominated Sorting Memetic Algorithm* (NSMA). We first show and describe the algorithmic scheme. Then, we formally introduce the *Front Multi-Objective Projected Gradient* (FMOPG) algorithm, which is the descent method used within the NSMA and for which we also provide a rigorous theoretical analysis.

3.1 Algorithmic scheme

The scheme of NSMA is reported in Algorithm 6. Basically, the structure of the proposed algorithm is similar to that of NSGA-II, from which we also inherit all the genetic operators. The main differences between the two methods are constituted by three new operations:

- `getSurrogateBounds` (Line 9);
- `getCrowdingDistanceThreshold` (Line 14);
- `optimizePopulation` (Line 17).

In the next subsections, we give a detailed description of these three new functions.

Algorithm 6: Non-dominated Sorting Memetic Algorithm

```

1 Input:  $F : \mathbb{R}^n \rightarrow \mathbb{R}^m$ ,  $\Omega = \{x \in \mathbb{R}^n | x \in [l, u]\}$  feasible set,  $X_0 \subset \Omega$ ,  $N$  population size,  $s_h \in \mathbb{R}_0^+$ ,
    $q \in [0, 1]$ ,  $n_{opt} \in \mathbb{N}^+$ ,  $\{\varepsilon_t\} \subset \mathbb{R}_0^+$  decreasing sequence.
2  $k = 0$ 
3  $t = 0$ 
4  $\hat{X}_0 = X_0$ 
5  $\hat{R}_0, \hat{C}_0 = \text{getMetrics}(\hat{X}_0)$ 
6  $X_0, R_0, C_0 = \text{getSurvivors}(\hat{X}_0, \hat{R}_0, \hat{C}_0, N)$ 
7 while a stopping criterion is not satisfied do
8    $P_k = \text{getParents}(X_k, R_k, C_k)$ 
9    $l_k^s, u_k^s = \text{getSurrogateBounds}(X_k, l, u, s_h)$ 
10   $O_k = \text{crossover}(P_k, l_k^s, u_k^s)$ 
11   $\tilde{O}_k = \text{mutation}(O_k, l_k^s, u_k^s)$ 
12   $\hat{X}_{k+1} = X_k \cup \tilde{O}_k$ 
13   $\hat{R}_{k+1}, \hat{C}_{k+1} = \text{getMetrics}(\hat{X}_{k+1})$ 
14   $\bar{c}_{k+1} = \text{getCrowdingDistanceThreshold}(\hat{X}_{k+1}, \hat{R}_{k+1}, \hat{C}_{k+1}, q)$ 
15   $X_{k+1}, R_{k+1}, C_{k+1} = \text{getSurvivors}(\hat{X}_{k+1}, \hat{R}_{k+1}, \hat{C}_{k+1}, N)$ 
16  if  $k \bmod n_{opt} = 0$  then
17     $X_{k+1}, R_{k+1}, C_{k+1} = \text{optimizePopulation}(F(\cdot), \Omega, X_{k+1}, R_{k+1}, C_{k+1}, \bar{c}_{k+1}, \varepsilon_t, N)$ 
18     $t = t + 1$ 
19   $k = k + 1$ 
20 return  $X_k$ 

```

3.1.1 Estimating surrogate bounds

When the addressed problem is characterized by a particularly large feasible region ($l \ll u$), the NSGA-II algorithm turns out to be slow at obtaining a good approximation of the Pareto front. This issue occurs because of the crossover and, above all, of the mutation operator. Random mutations over a large search area lead from the very first iterations to a population which is overly disperse and far from optimality. In such a scenario, even the effectiveness of the crossover operator might be compromised: some parents might have extremely bad features. As a consequence, NSGA-II may exhibit a performance slowdown.

In NSMA, this issue is solved using surrogate bounds for the crossover and the mutation operators instead of the original ones. These bounds are obtained using the getSurrogateBounds function, which we report in Algorithm 7. The surrogate bounds are computed using the current population and a shift value s_h . This latter parameter is employed to progressively enlarge the region where the population can be distributed: a greater value leads to a bigger enlargement.

Ideally, the exploration starts considering only a small portion of the feasible area, which is defined by the initial population and the shift value s_h . In this way, the points cannot be moved by the crossover and the mutation operators too far away in the feasible set. At each following iteration, new surrogate bounds are computed to enlarge the search space.

Algorithm 7: getSurrogateBounds

```

1 Input:  $X_k \subset \Omega, l, u \in \mathbb{R}^n$  lower and upper bounds,  $s_h \in \mathbb{R}_0^+$ .
2 for  $i = 1, \dots, n$  do
3    $(l_k^s)_i = \max \left\{ l_i, \min_{x \in X_k} \{x_i\} - s_h \right\}$ 
4    $(u_k^s)_i = \min \left\{ u_i, \max_{x \in X_k} \{x_i\} + s_h \right\}$ 
5 return  $l_k^s, u_k^s$ 

```

After a number of iterations, it may happen that the surrogate bounds cover a bigger region than the one defined by the original bounds. In such case, the search goes on over the entire feasible set.

3.1.2 Identifying exploration candidates

Similarly as in memetic approaches for scalar optimization, performing local searches starting from each point in a population usually turns out to be inefficient. In fact, a great computational effort is required to optimize many points that in the end do not lead to good solutions.

In the case of NSMA, one may think of only performing local searches for the rank-0 points. However, this idea is inefficient too: during the last iterations, most, if not all, the points are likely to be associated with a ranking value equal to 0. Furthermore, many of these points could be in a high density area of the Pareto front and, therefore, optimizing all of them could be a waste of computational time.

The issue is solved by choosing to optimize the rank-0 points associated with an high crowding distance. As already remarked in Sect. 2.2.1, such points are in a poorly populated area of the objectives space. Therefore, optimizing them, we still contribute to obtain a better approximation of the Pareto front, since they are rank-0 points, and, at the same time, we have the possibility to populate a low density area, leading to a better spread Pareto front.

Algorithm 8: getCrowdingDistanceThreshold

```

1 Input:  $\hat{X}_{k+1} \subset \Omega, \hat{R}_{k+1}, \hat{C}_{k+1} \in \mathbb{R}^{|\hat{X}_{k+1}|}$  metrics vectors,  $q \in [0, 1]$ .
2  $\bar{C}_{k+1} = \{\hat{c}_p \in \hat{C}_{k+1} | \hat{r}_p = 0 \wedge \hat{c}_p < +\infty\}$ 
3 if  $\bar{C}_{k+1} \neq \emptyset$  then
4   Let  $\bar{c}_{k+1}$  be the  $q$ -quantile of the set  $\bar{C}_{k+1}$ 
5 else
6    $\bar{c}_{k+1} = +\infty$ 
7 return  $\bar{c}_{k+1}$ 

```

Through the `getCrowdingDistanceThreshold` function, which we report in Algorithm 8, we retrieve the q -quantile of the crowding distances of the rank-0

points in \hat{X}_{k+1} . We denote by \bar{c}_{k+1} this quantity: only the rank-0 points associated with a crowding distance greater than or equal to \bar{c}_{k+1} will be optimized through the FMOPG algorithm. Smaller values for the parameter q lead to the optimization of a greater number of points.

As stated in [13], some points will be associated to a crowding distance equal to $+\infty$. These points are considered the extreme solutions of the Pareto front w.r.t. a specific objective function. For this reason, they are always used as starting solutions for local searches, since they could lead to a wider Pareto front approximation.

3.1.3 Local searches by multi-objective descent

In the `optimizePopulation` function, which we report in Algorithm 9, the FMOPG method is employed to refine the population by performing local searches. This function is the core of our memetic approach: it allows to combine the typical features of descent methods with the genetic operators of NSGA-II.

Algorithm 9: `optimizePopulation`

```

1 Input:  $F : \mathbb{R}^n \rightarrow \mathbb{R}^m$ ,  $\Omega$  feasible closed and convex set,  $X_{k+1} \subset \Omega$ ,  $R_{k+1}, C_{k+1} \in \mathbb{R}^{|X_{k+1}|}$ 
  metrics vectors,  $\bar{c}_{k+1}$  crowding distance threshold,  $\varepsilon_t \in \mathbb{R}_0^+$ ,  $N$  population size.
2  $\hat{X}_{k+1} = X_{k+1}$ 
3 for  $p = 1, \dots, |X_{k+1}|$  do
4   for  $I \in 2^{\{1, \dots, m\}}$  do
5     if  $x_p$  is such that
6       a.  $r_p = 0$ ,  $c_p \geq \bar{c}_{k+1}$ 
7       b.  $x_p \in \hat{X}_{k+1}^I$ 
8       c.  $\theta_{\Omega}^I(x_p) < 0$ 
9     then
10       $\tilde{X}_p = \text{FMOPG}(F(\cdot), \Omega, I, \hat{X}_{k+1}^I, x_p, \varepsilon_t)$ 
11       $\hat{X}_{k+1} = \hat{X}_{k+1} \cup \tilde{X}_p$ 
9  $\hat{R}_{k+1}, \hat{C}_{k+1} = \text{getMetrics}(\hat{X}_{k+1})$ 
10  $X_{k+1}, R_{k+1}, C_{k+1} = \text{getSurvivors}(\hat{X}_{k+1}, \hat{R}_{k+1}, \hat{C}_{k+1}, N)$ 
11 return  $X_{k+1}, R_{k+1}, C_{k+1}$ 

```

In order to be optimized through FMOPG w.r.t. a subset of indices of objectives $I \subseteq \{1, \dots, m\}$, a point x_p must satisfy the following conditions:

- Its rank must be 0 and its crowding distance must be greater than or equal to \bar{c}_{k+1} (Line 5a). These requirements are already discussed in Sect. 3.1.2.
- It must belong to \hat{X}_{k+1}^I , which is the set of mutually non-dominated points w.r.t. F_I contained in \hat{X}_{k+1} (Line 5b). A formal definition of this set can be found in Sect. 2.1. Trying to optimize points which are not contained in \hat{X}_{k+1}^I could be useless, since we have no guarantee to reach a non-dominated point w.r.t. F_I .
- It must not be Pareto-stationary w.r.t. F_I (Line 5c).

If the point satisfies all these requirements, it is used as starting solution in the FMOPG algorithm. Along with it, the set \hat{X}_{k+1}^I is used as input for the algorithm. FMOPG

returns the set of produced solutions, which are collected in the set \tilde{X}_p and inserted in the set \hat{X}_{k+1} .

Lastly, the new population \hat{X}_{k+1} is reduced in order to have exactly N survivors. This last operation is performed through the `getMetrics` (Sect. 2.2.1) and the `getSurvivors` (Sect. 2.2.3) functions of the NSGA-II algorithm.

3.2 The front multi-objective projected gradient algorithm

The *Front Multi-Objective Projected Gradient* (FMOPG) algorithm is the descent method used in our memetic approach. In particular, it is a variant of the MOPG method (Algorithm 3).

3.2.1 Algorithmic scheme

We report the scheme of FMOPG in Algorithm 10.

Algorithm 10: Front Multi-Objective Projected Gradient

```

1 Input:  $F : \mathbb{R}^n \rightarrow \mathbb{R}^m$ ,  $\Omega$  feasible closed and convex set,  $I \subseteq \{1, \dots, m\}$ ,  $X_0 \subset \Omega$ ,  $x_0 \in X_0$ .
2  $k = 0$ 
3 while  $x_k$  is not Pareto-stationary w.r.t.  $F_I$  do
4   Compute
      
$$\theta_{\Omega}^I(x_k) = \min_{z \in \Omega} \max_{j \in I} \nabla f_j(x_k)^T (z - x_k)$$

      
$$\|z - x_k\| \leq 1$$

5   Let  $d_{\Omega k}^I \in \mathcal{V}_{\Omega}^I(x_k)$  be the direction associated with  $\theta_{\Omega}^I(x_k)$ 
6    $\alpha_k = \text{B-FALS}(F(\cdot), \Omega, I, X_k^I, x_k, d_{\Omega k}^I, \theta_{\Omega}^I(x_k))$ 
7    $x_{k+1} = x_k + \alpha_k d_{\Omega k}^I$ 
8    $X_{k+1} = X_k \cup \{x_{k+1}\}$ 
9    $k = k + 1$ 
10 return sequence  $\{x_k\}$ 

```

The main difference between FMOPG and MOPG is the following: while in the original algorithm the current point x_k is only optimized w.r.t. itself, in FMOPG it is also w.r.t. the set of points in which it is contained. At each iteration, the constrained steepest descent direction $d_{\Omega k}^I$ at the current point x_k w.r.t. the subset of indices of the objectives I is found (Line 5). Then, in Line 6 a step size α_k is calculated by the B-FALS procedure (Algorithm 4). Given the direction and the step size, a new point x_{k+1} is obtained (Line 7). This latter one is inserted in the set X_k , leading to a new set X_{k+1} (Line 8).

The FMOPG algorithm iterates until the current solution x_k is Pareto-stationary w.r.t. F_I . At the end, the method returns the sequence of points $\{x_k\}$ generated during the iterations. Indeed, considering the stopping conditions of B-FALS, we have no guarantee that, for all k , the point x_{k+1} dominates x_k w.r.t. F_I . So, every point produced by FMOPG could be useful to obtain good and spread Pareto front approximations.

Finally, note that the FMOPG algorithm is called by the `optimizePopulation` function with an additional parameter ε_t (Line 7 of Algorithm 5). In fact, FMOPG is executed using ε -Pareto-stationarity as stopping condition. In NSMA (Algorithm 6), we consider a decreasing sequence $\{\varepsilon_t\} \subset \mathbb{R}_0^+$. So, during the iterations, we get closer and closer to the Pareto-stationarity.

3.2.2 Algorithm analysis

In this section, we provide a rigorous analysis of the FMOPG algorithm from a theoretical perspective. The following analysis is crucial to state the convergence properties of FMOPG. These latter ones are crucial to guarantee that local searches within NSMA stop in finite time and, thus, the overall algorithm is well-defined.

Before proceeding, we need to state an assumption.

Assumption 1 Let $I \subseteq \{1, \dots, m\}$, $X_0 \subset \Omega$ be a set of feasible points and $x_0 \in X_0$. There does not exist a point $y_0 \in X_0$ that dominates x_0 w.r.t. F_I , i.e. $x_0 \in X_0^I$.

This assumption is reasonable since a point x_p to be optimized through FMOPG must be non-dominated w.r.t. F_I (Sect. 3.1.3).

We begin characterizing the points produced by the FMOPG algorithm.

Proposition 2 Consider a generic iteration k of FMOPG. Let $I \subseteq \{1, \dots, m\}$, X_k be a set of feasible points and $x_k \in X_k$. Assume that x_k is not dominated by any point in X_k w.r.t. F_I . Then, B-FALS returns a step size $\alpha_k > 0$ such that the point $x_{k+1} = x_k + \alpha_k d_{\Omega k}^I$ is feasible and not dominated by any point in X_{k+1} w.r.t. F_I .

Proof The B-FALS algorithm is performed from $x_k \in X_k^I$, with $\theta_{\Omega}^I(x_k) < 0$, along a constrained steepest descent direction $d_{\Omega k}^I$. Then, from Proposition 1, B-FALS terminates in a finite number of steps and returns a step size $\alpha_k > 0$ such that the point $x_{k+1} = x_k + \alpha_k d_{\Omega k}^I$ has the following properties:

- $x_{k+1} \in \Omega$;
- x_{k+1} is not dominated by any other point in X_k w.r.t. F_I .

Since $X_{k+1} = X_k \cup \{x_{k+1}\}$, the assertion is finally proved. □

Remark 1 Since the point x_{k+1} induced by the step size produced by B-FALS is not dominated by any point in X_{k+1} w.r.t. F_I , we can easily conclude that the new point is also not dominated w.r.t. all the objectives.

Given Proposition 2, we can state the following corollary.

Corollary 1 Let Assumption 1 hold with $I \subseteq \{1, \dots, m\}$, the set X_0 and the point x_0 . Then, the sequence of sets $\{X_k\}$ and the sequence of points $\{x_k\}$ generated by FMOPG are such that for all $k = 0, 1, \dots$, x_k is feasible and not dominated by any point in X_k w.r.t. F_I .

Proof The assertion straightforwardly follows if the assumptions of Proposition 2 are satisfied at every iteration k of the algorithm.

When $k = 0$, this is guaranteed by Assumption 1. The case of a generic iteration k simply follows by induction from Proposition 2 itself. □

Before proceeding with the convergence analysis, we make a further reasonable assumption. This hypothesis is similar to Assumption 1 in [10]. However, in this context, Assumption 1 and bound constraints must be also taken into account.

Assumption 2 Assumption 1 holds and x_0 is also such that:

- x_0 is not Pareto-stationary w.r.t. F_I ;
- the set $\mathcal{L}(x_0) = \bigcup_{j=1}^m \{x \in \Omega : f_j(x) \leq f_j(x_0)\}$ is compact.

This assumption is stronger than the one required to prove convergence of the MOSD method (Lemma 2). However, as also observed in [10] for FALS (Algorithm 2), this is reasonable since the second stopping criterion of B-FALS is weaker than the one used in ALS (Algorithm 1).

Proposition 3 *Let Assumption 2 hold with $I \subseteq \{1, \dots, m\}$, the set X_0 and the point x_0 . Let $\{x_k\}$ be the sequence of points generated by FMOPG. Then $\{x_k\}$ admits limit points and every limit point is Pareto-stationary considering the objectives f_j , with $j \in I$.*

Proof Firstly, we prove that the sequence $\{x_k\}$ admits limit points. Since $x_0 \in X_k$ for all k , Corollary 1 guarantee that, for each k , $x_k \in \Omega$ and there exists an index $j(x_k) \in I$ such that

$$f_{j(x_k)}(x_k) \leq f_{j(x_k)}(x_0).$$

So,

$$x_k \in \{x \in \Omega : f_{j(x_k)}(x) \leq f_{j(x_k)}(x_0)\}$$

and, therefore,

$$x_k \in \mathcal{L}(x_0), \quad \forall k.$$

Assumption 2 assures that the sequence $\{x_k\}$ is bounded. Hence, this latter one admits limit points: we can consider a subsequence $K \subseteq \{1, 2, \dots\}$ such that

$$\lim_{\substack{k \rightarrow \infty \\ k \in K}} x_k = \bar{x}.$$

We recall that \bar{x} is Pareto-stationary w.r.t. F_I if and only if $\theta_\Omega^I(\bar{x}) = 0$. By contradiction, we assume that \bar{x} is not Pareto-stationary w.r.t. F_I : there exists $\bar{\varepsilon} > 0$ such that

$$\theta_\Omega^I(x_k) \leq -\bar{\varepsilon} < 0, \quad \forall k \in K. \tag{8}$$

Next, we want to prove the following statement:

$$\lim_{\substack{k \rightarrow \infty \\ k \in K}} \alpha_k \theta_\Omega^I(x_k) = 0. \tag{9}$$

Again, by contradiction, we assume that the assertion is not true: there exists a subsequence $\bar{K} \subseteq K$ and $\bar{\eta} > 0$ such that

$$\alpha_k \theta_{\Omega}^I(x_k) \leq -\bar{\eta} < 0, \quad \forall k \in \bar{K}. \tag{10}$$

Recalling Proposition 1 and Corollary 1, for all $k \in \bar{K}$, B-FALS returns in a finite number of iterations a step size α_k such that

$$x_{k+1} = x_k + \alpha_k d_{\Omega k}^I \in \Omega \tag{11}$$

and

$$F_I(y_k) + \mathbf{1}\beta\alpha_k\theta_{\Omega}^I(x_k) \not\leq F_I(x_{k+1}),$$

for all $y_k \in X_k$. By using Eq. (10), we obtain that, for all $k \in \bar{K}$ and for all $y_k \in X_k$,

$$F_I(y_k) - \mathbf{1}\beta\bar{\eta} \not\leq F_I(x_{k+1}). \tag{12}$$

Since $\beta > 0$ and $\bar{\eta} > 0$, we have that $-\beta\bar{\eta} < 0$.

Since $X_k = X_0 \cup \{x_1\} \cup \dots \cup \{x_k\}$ and $x_0 \in X_0$, it simply follows that, for all $k \in \bar{K}$,

$$F_I(x_0) - \mathbf{1}\beta\bar{\eta} \not\leq F_I(x_{k+1}).$$

Therefore, for all $k \in \bar{K}$, there exists $j_k \in I$ such that

$$f_{j_k}(x_0) > f_{j_k}(x_0) - \beta\bar{\eta} \geq f_{j_k}(x_{k+1}),$$

and, then, considering also Eq. (11),

$$x_{k+1} \in \mathcal{L}(x_0). \tag{13}$$

Moreover, let us consider $k_1, k_2 \in \bar{K}$, with $k_1 < k_2$. By the instructions of the algorithm, we know that $x_{k_1+1} \in X_{k_2}$. Thus, from Eq. (12), we know that

$$F_I(x_{k_1+1}) - \mathbf{1}\beta\bar{\eta} \not\leq F_I(x_{k_2+1})$$

Therefore, for any pair $k_1, k_2 \in \bar{K}$, with $k_1 < k_2$, there exists $j_{k_2} \in I$ such that

$$f_{j_{k_2}}(x_{k_1+1}) - \beta\bar{\eta} \geq f_{j_{k_2}}(x_{k_2+1}). \tag{14}$$

Equations(13) and (14) imply that we have an infinite sequence $\{F_I(x_{k+1})\}_{k \in \bar{K}}$, with $x_{k+1} \in \mathcal{L}(x_0)$, where any pair of points is at a distance not smaller than $\beta\bar{\eta}$ from each other. Thus, the set

$$Z = \{z \in \mathbb{R}^m \mid z = F(x_{k+1}), x_{k+1} \in \mathcal{L}(x_0), k \in \bar{K}\}$$

is not compact. This last statement and the continuity of F contradict Assumption 2, since the image of a compact set under a continuous map should be compact. Thus, Equation (9) holds.

Recalling Eq. (8), from Equation (9) we obtain the following statement:

$$\lim_{\substack{k \rightarrow \infty \\ k \in K}} \alpha_k = 0.$$

Given this limit, we can consider sufficiently large values for $k \in K$ such that

$$\alpha_k < \frac{\alpha_k}{\delta} \leq 1. \tag{15}$$

Since Ω is convex and $d_{\Omega k}^I$ is a feasible direction by construction, Eq. (15) implies that the point $x_k + (\alpha_k/\delta)d_{\Omega k}^I \in \Omega$. Therefore, for sufficiently large values for $k \in K$, the stopping conditions of B-FALS imply that there exists a point $y_k \in X_k$ such that

$$F_I(y_k) + \mathbf{1}\beta \frac{\alpha_k}{\delta} \theta_{\Omega}^I(x_k) < F_I\left(x_k + \frac{\alpha_k}{\delta} d_{\Omega k}^I\right). \tag{16}$$

Considering Corollary 1 and Eq. (16) respectively, we have that an index $j(x_k) \in I$ exists such that

$$f_{j(x_k)}(x_k) + \beta \frac{\alpha_k}{\delta} \theta_{\Omega}^I(x_k) \leq f_{j(x_k)}(y_k) + \beta \frac{\alpha_k}{\delta} \theta_{\Omega}^I(x_k)$$

and

$$f_{j(x_k)}(y_k) + \beta \frac{\alpha_k}{\delta} \theta_{\Omega}^I(x_k) < f_{j(x_k)}\left(x_k + \frac{\alpha_k}{\delta} d_{\Omega k}^I\right).$$

Since the set I is finite, we can consider a subsequence $\bar{K} \subseteq K$ such that, for sufficiently large values for $k \in \bar{K}$, $j(x_k) = \hat{j}$ and, combining the two above inequalities,

$$f_{\hat{j}}\left(x_k + \frac{\alpha_k}{\delta} d_{\Omega k}^I\right) - f_{\hat{j}}(x_k) > \beta \frac{\alpha_k}{\delta} \theta_{\Omega}^I(x_k).$$

Using the Mean-value Theorem, we have that

$$f_{\hat{j}}\left(x_k + \frac{\alpha_k}{\delta} d_{\Omega k}^I\right) - f_{\hat{j}}(x_k) = \frac{\alpha_k}{\delta} \nabla f_{\hat{j}}(\xi_k)^T d_{\Omega k}^I,$$

with

$$\xi_k = x_k + t_k \frac{\alpha_k}{\delta} d_{\Omega k}^I, \quad t_k \in (0, 1).$$

Then, we can write

$$\nabla f_{\hat{j}}(\xi_k)^T d_{\Omega k}^I > \beta \theta_{\Omega}^I(x_k),$$

from which we can state that

$$\nabla f_{\hat{j}}(x_k)^T d_{\Omega k}^I + \left[\nabla f_{\hat{j}}(\xi_k) - \nabla f_{\hat{j}}(x_k) \right]^T d_{\Omega k}^I > \beta \theta_{\Omega}^I(x_k).$$

Since $\hat{j} \in I$, we have that

$$\theta_{\Omega}^I(x_k) = \max_{j \in I} \nabla f_j(x_k)^T d_{\Omega k}^I \geq \nabla f_{\hat{j}}(x_k)^T d_{\Omega k}^I$$

and

$$(1 - \beta)\theta_{\Omega}^I(x_k) + \left[\nabla f_{\hat{j}}(\xi_k) - \nabla f_{\hat{j}}(x_k) \right]^T d_{\Omega k}^I > 0.$$

Using Eq. (8), we obtain

$$-(1 - \beta)\bar{\varepsilon} + \left[\nabla f_{\hat{j}}(\xi_k) - \nabla f_{\hat{j}}(x_k) \right]^T d_{\Omega k}^I > 0.$$

By taking the limit for $k \rightarrow \infty, k \in \bar{K}$, recalling the continuity of J_F , the boundedness of $d_{\Omega k}^I$ and that $\alpha_k \rightarrow 0$, we get that

$$-(1 - \beta)\bar{\varepsilon} > 0.$$

Since $1 - \beta > 0$ and $\bar{\varepsilon} > 0$, we get the contradiction. So, we prove that the limit point \bar{x} of the sequence $\{x_k\}$ is Pareto-stationary w.r.t. F_I . \square

Finally, we prove that, when a stopping criterion based on the ε -Pareto-stationarity is considered, FMOPG is well defined, i.e., it terminates in a finite number of iterations.

Proposition 4 *Let Assumption 2 hold with $I \subseteq \{1, \dots, m\}$, the set X_0 and the point x_0 . Let $\varepsilon > 0$. Then, the FMOPG algorithm finds in a finite number of steps a point x_k which is ε -Pareto-stationary w.r.t. F_I .*

Proof We assume, by contradiction, that FMOPG produces an infinite sequence of points $\{x_k\}$ such that, for all k , x_k is not ε -Pareto-stationary w.r.t. F_I . Since Assumption 2 holds, Proposition 3 ensures that there exists a subsequence $K \subseteq \{1, 2, \dots\}$ such that

$$\lim_{\substack{k \rightarrow \infty \\ k \in K}} x_k = \bar{x}$$

and \bar{x} is Pareto-stationary w.r.t. F_I , i.e., recalling Definition 3, for all $z \in \Omega$ such that $\|z - \bar{x}\| \leq 1$ we have that

$$\max_{j \in I} \nabla f_j(\bar{x})^T (z - \bar{x}) \geq 0.$$

Given the continuity of the max operator and J_F , we can state that

$$\lim_{\substack{k \rightarrow \infty \\ k \in K}} \max_{j \in I} \nabla f_j(x_k)^T (z - x_k) \geq 0 > -\varepsilon.$$

This last statement implies that, for sufficiently large values for $k \in K$, for all $z \in \Omega$ such that $\|z - x_k\| \leq 1$, the following equation has to hold:

$$\max_{j \in I} \nabla f_j(x_k)^T (z - x_k) > -\varepsilon,$$

i.e. x_k is ε -Pareto-stationary w.r.t. F_I . Therefore, we get the contradiction and the assertion is proved. \square

4 Computational experiments

In this section, we provide the results of thorough computational experiments, focusing on the comparison of NSMA with the main state-of-the-art methods in diverse settings. The code of all the algorithms was written in Python3 ¹. In addition, all the tests were run on a computer with the following characteristics: Ubuntu 20.04, Intel Xeon Processor E5-2430 v2 6 cores 2.50 GHz, 16 GB RAM. We used the Gurobi Optimizer (Version 9) to solve instances of Problem (6).

4.1 Settings

In this section, we report detailed information on the settings used for all the considered algorithms in our experiments, the metrics and the problems used to carry out the comparison.

4.1.1 Metrics

In this section, we provide a little description of the metrics and tools used to compare the algorithms.

The first three metrics are the ones introduced in [12]: *purity*, Γ -*spread* and Δ -*spread*. These metrics are widely used to evaluate the performance of multi-objective optimization algorithms.

We recall that the *purity* metric measures the quality of the generated front, i.e., how effective a solver is at obtaining non-dominated points w.r.t. its competitors. In detail, the purity value indicates the ratio of the number of non-dominated points that a solver obtained over the number of the points produced by that solver. Clearly, a higher value is related to a better performance. In order to calculate the *purity* metric, we need a reference front to establish whether a point is dominated or not. In our experiments, we

¹ The implementation code of the NSMA algorithm can be found at <https://github.com/pierlumanzu/nsma> [42].

considered as the reference front the one obtained by combining the fronts retrieved by all the considered algorithms and by discarding the dominated points.

The *spread* metrics are equally essential, since they measure the uniformity of the generated fronts in the objectives space. The Γ -*spread* is defined as the maximum ℓ_∞ distance in the objectives space between adjacent points of the Pareto front, while the Δ -*spread* basically measures the standard deviation of the ℓ_∞ distance between adjacent Pareto front points. As opposed to the *purity*, low values for the *spread* metrics are associated with good performance.

In addition to the previous metrics, we used the *ND-points* metric, introduced in [9]. This score substantially indicates the number of non-dominated points obtained by a solver w.r.t. the reference front. We consider this metric as important as the *purity* one: in particular, we think that these two metrics should be considered complementary.

Lastly, we employed the performance profiles introduced in [14] to carry out the comparison. Performance profiles are a useful tool to appreciate the relative performance and robustness of the considered algorithms. The performance profile of a solver w.r.t. a certain metric is the (cumulative) distribution function of the ratio of the score obtained by a solver over the best score among those obtained by all the considered solvers. In other words, it is the probability that the score achieved by a solver in a problem is within a factor $\tau \in \mathbb{R}$ of the best value obtained by any of the solvers in that problem. For a more technical explanation, we refer the reader to [14]. Note that performance profiles w.r.t. *purity* and *ND-points* were produced based on the inverse of the obtained values, since the metrics have increasing values for better solutions.

4.1.2 Algorithms and hyper-parameters

The first two algorithms we chose for the comparisons are, naturally, the NSGA-II [13] and the FPGA [10] procedures, described in Sect. 2.2 and Appendix A, respectively. We consider these methods as representatives for EAs and descent methods and, thus, NSMA most direct competitors. The parameters values for both algorithms were chosen according to the reference papers. Like NSMA, in NSGA-II the number N of solutions in the population was fixed to 100.

Then, the values for the parameters of NSMA were chosen based on some preliminary experiments on a subset of the tested problems, which we do not report here for the sake of brevity. The values are:

- $N = 100$;
- $s_h = 10$;
- $q = 0.9$;
- $n_{opt} = 5$;
- in B-FALS $\alpha_0 = 1$, $\beta = 10^{-4}$, $\delta = 0.5$.

We also consider in the experiments the DMS algorithm [12], a multi-objective derivative-free method, inspired by the search/poll paradigm of direct-search methodologies of directional type. DMS maintains a list of non-dominated points, from which the new iterates or poll centers are chosen. The parameters for this method were set

Table 1 Problems used in the computational experiments

PROBLEM	n	m
CEC09_1, CEC09_2, CEC09_3,	5, 10, 20, 30,	2
CEC09_4, CEC09_5, CEC09_6, CEC09_7	40, 50, 100, 200	
CEC09_8, CEC09_9, CEC09_10	5, 10, 20, 30,	3
	40, 50, 100, 200	
ZDT_1, ZDT_2, ZDT_3, ZDT_4	2, 5, 10, 20, 30,	2
	40, 50, 100, 200	
MOP_1	1	2
MOP_2	2, 5, 10, 20, 30,	2
	40, 50, 100, 200	
MOP_3	2	2
MAN	2, 5, 10, 20, 30,	2
	40, 50, 100, 200	

according to the reference paper and the code available online (<http://www.mat.uc.pt/dms>).

In most of the computational experiments, for each algorithm and problem we ran the test for up to 2 minutes. A stopping criterion based on a time limit is the fairest way to compare such structurally different algorithms. Obviously, we also took into account other specific stopping criteria indicating that a certain algorithm cannot improve the solutions anymore.

NSMA and NSGA-II are non-deterministic algorithm. Therefore, we decided to run them 5 times on every problem, with different seeds for the pseudo-random number generator. Every execution was characterized by the same time limit (2 minutes). The five generated fronts were compared based on the *purity* metric and only the best one was chosen as the output of NSMA/NSGA-II. In this context, the reference front was the combination of the fronts of the 5 executions. Executing 5 runs lets NSMA/NSGA-II reduce its sensibility to the seed used for its random operations. On the other side, FPGA and DMS are deterministic and, then, they were executed once.

4.1.3 Problems

The problems constituting the benchmark of the computational experiments are listed in Table 1. In this benchmark, we considered problems whose objective functions are at least continuously differentiable almost everywhere. If a problem is characterized by singularities, we counted these latter ones as Pareto-stationary points. All the constraints are defined by finite lower and upper bounds.

The set is mainly composed by the CEC09 problems [56], the ZDT problems [57] and the MOP problems [32]. In particular, some of the CEC09 and the ZDT problems have particularly difficult objective functions. Hence, these problems are particularly interesting for the analysis of the behavior of the algorithms with hard tasks.

We also defined a new test problem with convex objective functions: we refer to it as the MAN problem. Its formal definition is the following:

$$\begin{aligned} f_1(x) &= \sum_{i=1}^n (x_i - i)^2 / n^2 \\ \min_{x \in \mathbb{R}^n} \quad & f_2(x) = \sum_{i=1}^n e^{-x_i} + x_i \\ \text{s.t.} \quad & x \in [-10^4, 10^4]. \end{aligned}$$

Inspired by Custódio et al. [12], for each problem the initial points were uniformly selected from the hyper-diagonal defined by the bound constraints. Furthermore, the number of initial points is equal to the dimension n of the problem. Since in the MOP_1 problem $n = 1$, only in this case we started the tests from one feasible point, namely, $x = 0$.

4.2 Experimental comparisons between NSGA-II and FPGA

Before turning to the evaluation of the NSMA, we carry out a preliminary study.

Evolutionary algorithms and descent methods have their own drawbacks. In particular, EAs do not have theoretical convergence properties. In addition, they can be very expensive in particular settings. On the other side, descent algorithms suffer on highly non-convex problems: in these cases, they often produce sub-optimal solutions, especially when the starting points are not chosen carefully.

In this section, we want to address two topics:

- the impact of convexity of the objective functions on the performance of these algorithms;
- the behavior of the methods as the problem dimension n increases.

For the comparisons of this section, we only considered the NSGA-II and FPGA algorithms, which we respectively pick as representatives for the two classes of methods.

As benchmark, we picked four problems that are scalable w.r.t. the problem dimension n and have the following features:

- the MAN problem and the ZDT_1 problem have convex objective functions;
- the CEC09_4 problem and the ZDT_3 problem have nonconvex objective functions.

For these comparisons, each problem was tested for values of $n \in \{5, 10, 20, 30, 40, 50, 100, 200\}$.

We show the performance profiles for the two algorithms on the convex problems in Fig. 1 and on the nonconvex problems in Fig. 2.

We can observe that in the former case the FPGA turned out to be better than NSGA-II in terms of *purity*. This result reasonably comes from the fact that, in problems characterized by convex objective functions, the use of first-order information and common descent directions lets FPGA find better solutions than NSGA-II for equal

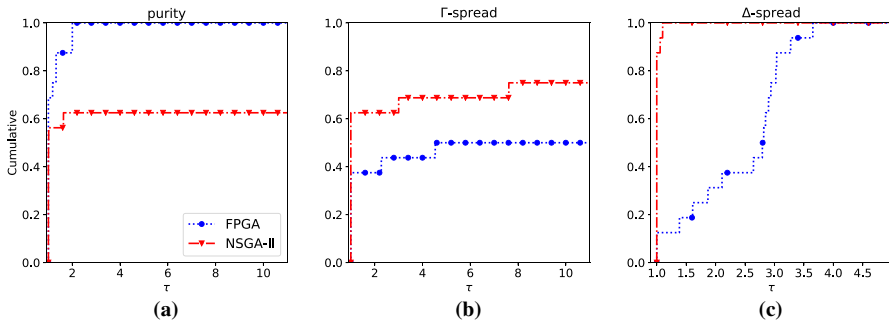


Fig. 1 Performance profiles for FPGA and NSGA-II on the *convex* MAN and ZDT_1 problems (for interpretation of the references to color in text, the reader is referred to the web version of the article). **a** Purity. **b** Γ -spread. **c** Δ -spread

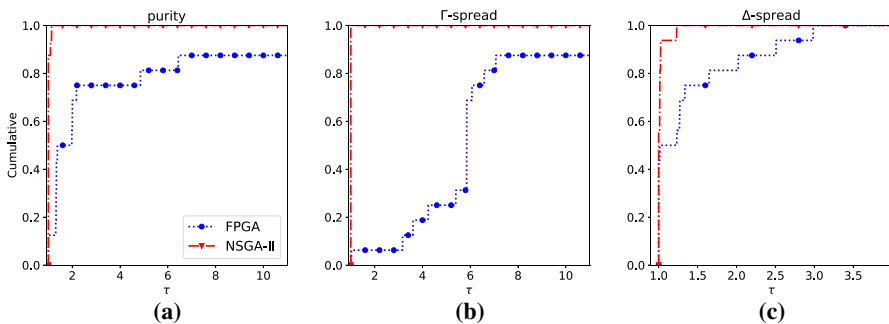


Fig. 2 Performance profiles for FPGA and NSGA-II on the *nonconvex* CEC09_4 and ZDT_3 problems (for interpretation of the references to color in text, the reader is referred to the web version of the article). **a** Purity. **b** Γ -spread. **c** Δ -spread

computational budget. On the contrary, the FPGA algorithm was outperformed by NSGA-II in terms of Γ -spread and Δ -spread. In this perspective, the crossover and mutation operations of NSGA-II allow to consistently obtain spread Pareto front approximations, while the constrained steepest partial descent directions and the B-FALS employed by the FPGA are apparently not as effective.

As for the nonconvex case, we can observe from the *purity* profile that now the FPGA obtained many points that are dominated by those produced by NSGA-II. The results with the *spread* metrics are instead analogous to the convex case, with NSGA-II outperforming FPGA. However, the performance gap in terms of Γ -spread is even larger, while it is less marked for the Δ -spread.

In order to assess the performance of the algorithms as the problem dimension n increases, in Tables 2 and 3 we show in detail the metrics values achieved by the two methods on a convex (MAN) and a nonconvex (CEC09_4) problems. Again, the table shows the overall strength of NSGA-II w.r.t. the Δ -spread metric.

As for the Γ -spread, in the MAN problem we can observe the great results achieved by FPGA: it outperformed NSGA-II considering values of n equal to or greater than 20. In these cases, the constrained steepest partial descent directions and the B-FALS algorithm turned out to be helpful in exploring the extreme regions of the objectives

Table 2 Metrics values obtained by FPGA and NSGA-II in the MAN problem with $n = 5, 10, 20, 30, 40, 50, 100, 200$

n	MAN (F convex)					
	<i>Purity</i>		Γ -spread		Δ -spread	
	FPGA	NSGA-II	FPGA	NSGA-II	FPGA	NSGA-II
5	0.984	0.98	1.839	0.809	1.928	0.69
10	0.993	0.61	6.241	1.377	1.791	0.547
20	1.0	0.05	16.65	49.983	1.59	0.754
30	1.0	0.0	2.318	89.577	1.353	0.724
40	1.0	0.0	36.716	279.445	1.458	0.906
50	1.0	0.0	5.868	412.791	1.231	0.889
100	1.0	0.0	21.854	3894.709	0.95	1.004
200	1.0	0.0	62.971	9283.624	0.824	0.906

The values marked in bold are the best obtained in a specific problem. Each of them is related to a specific score

Table 3 Metrics values obtained by FPGA and NSGA-II in the CEC09_4 problem with $n = 5, 10, 20, 30, 40, 50, 100, 200$

n	CEC09_4 (F non-convex)					
	<i>purity</i>		Γ -spread		Δ -spread	
	FPGA	NSGA-II	FPGA	NSGA-II	FPGA	NSGA-II
5	0.0	1.0	0.419	0.132	0.579	0.713
10	0.154	0.99	0.607	0.041	1.109	0.549
20	0.714	0.98	0.475	0.037	0.68	0.537
30	0.429	0.93	0.553	0.078	0.678	0.55
40	0.2	0.97	0.501	0.093	0.786	0.618
50	0.025	0.95	0.544	0.128	1.845	0.618
100	0.965	0.89	0.514	0.143	1.757	0.701
200	0.915	0.81	0.483	0.474	1.673	1.016

The values marked in bold are the best obtained in a specific problem. Each of them is related to a specific score

space and, then, in finding a spread approximation of the Pareto front. In the CEC09_4 problem, it is the opposite: the genetic algorithm managed to obtain the best Γ -spread values.

The *purity* values indicate another relevant feature of the two algorithms. In the nonconvex case, FPGA turned out not to be capable of obtaining better points than NSGA-II for low values of n . However, as the value of n increased, the situation gradually changed and FPGA finally obtained better *purity* values w.r.t. its competitor on the largest problems. These results remark one of the drawbacks of the EAs, i.e., the limited scalability. In this case, common descent directions can be very helpful for cheaply improving the quality of the solutions.

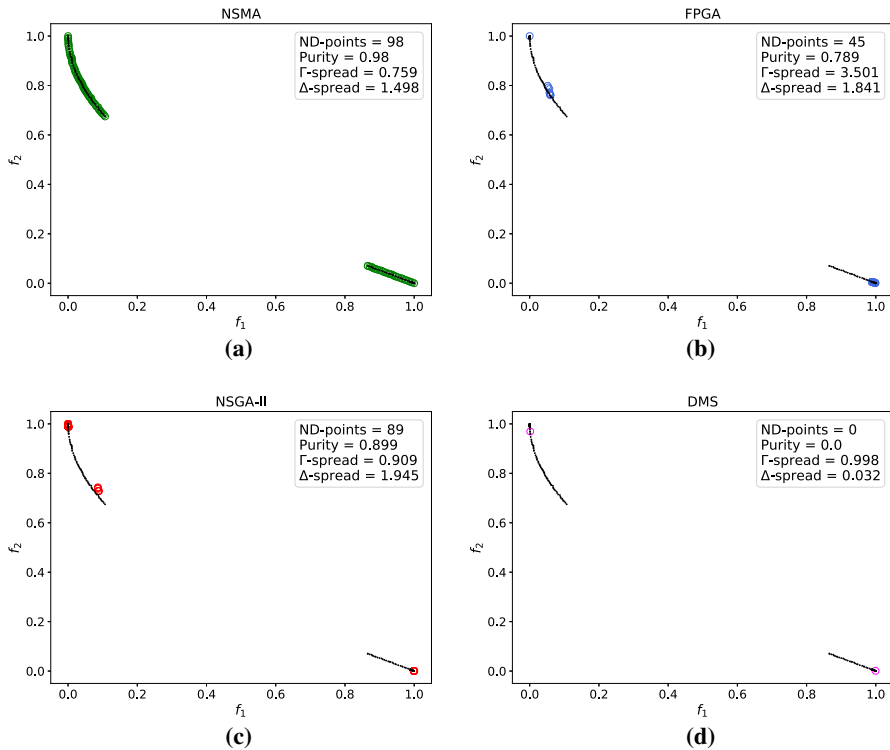


Fig. 3 Approximation of the Pareto front of the CEC09_3 problem with $n = 10$ (for interpretation of the references to color in text, the reader is referred to the electronic version of the article). **a** NSMA. **b** FPGA. **c** NSGA-II. **d** DMS

In conclusion, both algorithms have features that make them very effective in specific situations: FPGA was better in convex and/or high dimensional problems, while NSGA-II was more effective in non-convex low dimensional ones. Furthermore, the genetic features of NSGA-II let this latter one perform better in finding spread and uniform Pareto fronts most of the times: this is also reflected in the *spread* metrics values obtained by NSGA-II. All these facts remark once again how much trying to join these benefits in one algorithm might be appealing.

4.3 Preliminary comparisons between NSMA and the state-of-the-art algorithms

In this section, we provide the results on two problems along with some first comments about the behavior of the four algorithms. We analyzed the CEC09_3 problem with $n = 10$ and the ZDT_3 problem with $n = 20$. The first one has particularly difficult objective functions, while the second one is also characterized by a composite function and a disconnected front which is not convex everywhere. We consider these problems suitable to start an analysis about the performance of the considered algorithms.

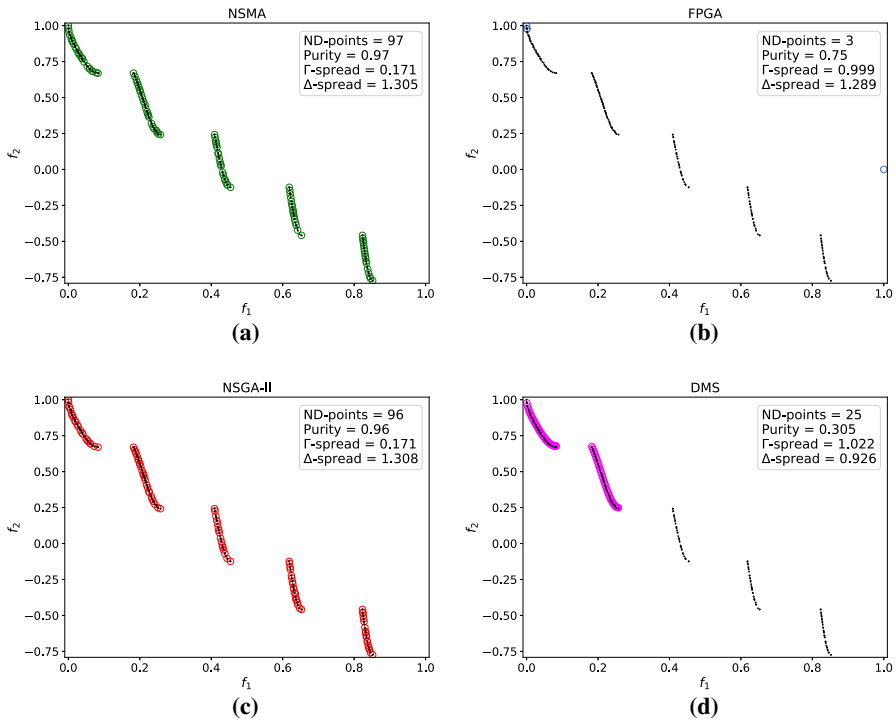


Fig. 4 Approximation of the Pareto front of the ZDT₃ problem with $n = 20$ (for interpretation of the references to color in text, the reader is referred to the electronic version of the article). **a** NSMA. **b** FPGA. **c** NSGA-II. **d** DMS

From the results on the CEC09₃ problem, shown in Fig. 3, we immediately observe the effectiveness of our approach. Indeed, NSMA outperformed the other algorithms in terms of *ND-points*, *purity* and Γ -*spread*.

NSGA-II and FPGA turned out to be the second and the third best algorithms, respectively, with FPGA outperforming the genetic method only in terms of Δ -*spread*. Note that FPGA achieved a high value for the Γ -*spread* metric since it produced a suboptimal point that is dominated and far from the reference front. This point is not shown in the figure for graphical reasons.

NSGA-II and FPGA seem not to be capable of spreading the search in the objectives space. Indeed, they retrieved many points but most of them are concentrated in a small portion of the objectives space. In this regard, NSMA was better: this result arguably comes from the use of constrained steepest partial descent directions with points characterized by a high crowding distance. Indeed, using descent steps at such points lets NSMA obtain a more spread and uniform Pareto front approximation w.r.t. its competitors.

NSMA and NSGA-II turned out to be the best algorithms on the ZDT₃ problem, as we can observe in Fig. 4. Furthermore, they exhibited very similar performance. It is known that NSGA-II is one of the most effective algorithms to use with the ZDT problem class. Indeed, its genetic features allow it to escape from non-optimal

Table 4 Metrics values achieved by the four algorithms (NSMA, FPGA, NSGA-II and DMS) on the convex MAN problem for $n = 5, 20, 50, 100$

n	METRIC	NSMA	FPGA	NSGA-II	DMS
5	<i>ND-points</i>	9	1107	2	5291
	<i>purity</i>	0.09	0.954	0.02	0.994
	Γ -spread	0.433	1.839	0.81	0.006
	Δ -spread	0.597	1.928	0.69	1.003
20	<i>ND-points</i>	59	2649	0	0
	<i>purity</i>	0.59	0.99	0.0	0.0
	Γ -spread	6.535	68.252	51.601	9953.2
	Δ -spread	0.558	1.541	0.775	N/A
50	<i>ND-points</i>	30	2776	0	0
	<i>purity</i>	0.3	0.998	0.0	0.0
	Γ -spread	46.41	273.518	543.112	9790.081
	Δ -spread	0.509	1.222	0.893	N/A
100	<i>ND-points</i>	22	2258	0	0
	<i>purity</i>	0.22	0.999	0.0	0.0
	Γ -spread	278.269	1154.101	3894.709	9878.008
	Δ -spread	0.521	0.952	1.005	N/A

The values marked in bold are the best obtained on a specific problem

Pareto-stationary solutions and to obtain good results with the most complex functions. NSMA seems to use these features as efficiently as NSGA-II. We also observe a little performance enhancement in terms of *ND-points* and *purity*.

The lack of these characteristics did not allow FPGA to have the same performance. Indeed, although this algorithm obtained a good value for the *purity* metric, it produced few points and it was not capable to obtain a spread and uniform Pareto front. DMS seems not to have the same issues, having been able to properly identify two blocks of the disconnected front. However, it performed worse than FPGA in terms of *purity*.

Finally, we note that NSMA was better than all its competitors in terms of *ND-points*. It was not obvious a priori to obtain such results, since, as opposed to FPGA and DMS, NSMA considers a fixed number of solutions in the population.

4.4 Performance analysis in variable settings

In this section, we want to assess the robustness of the proposed algorithm in the specific settings where, as highlighted in Sect. 4.2, genetic and descent methods exhibit particular struggles. In detail, we compare the performance of the four algorithms (NSMA, FPGA, NSGA-II and DMS) in two peculiar problems already addressed in Sect. 4.2: MAN (F convex) and CEC09_4 (F nonconvex). Moreover, we consider the following problem dimensionalities: $n = 5, 20, 50, 100$.

The results for the MAN problem are shown in Fig. 5 and Table 4. For $n = 5$, DMS turned out to be the best algorithm in all the metrics except for the Δ -spread. Only

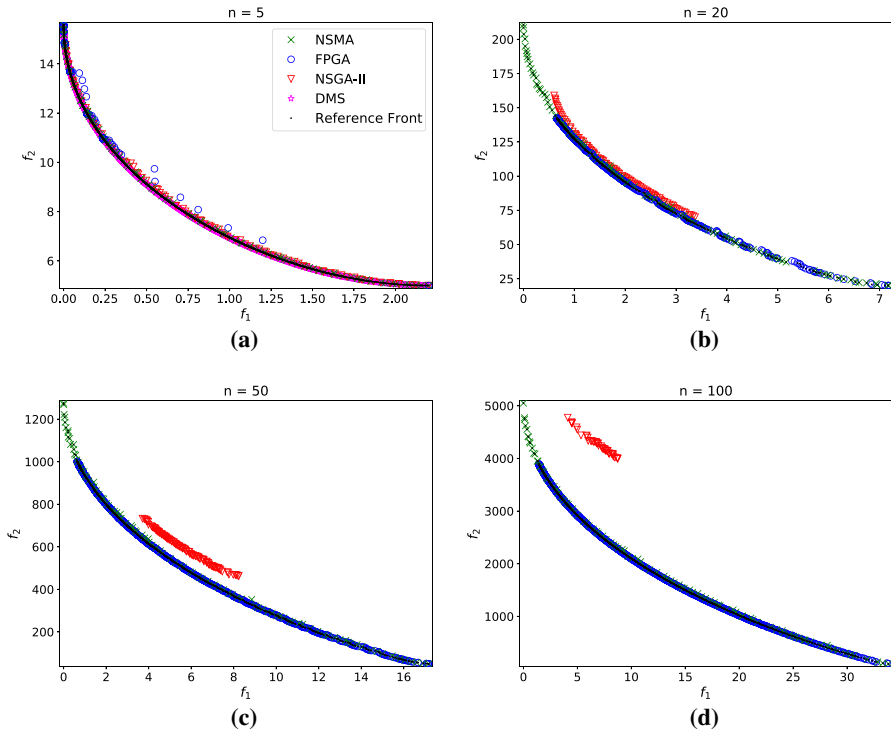


Fig. 5 Approximation of the Pareto front of the convex MAN problem at different dimensionalities, retrieved by NSMA, FPGA, NSGA-II and DMS (for interpretation of the references to color in text, the reader is referred to the electronic version of the article). **a** $n = 5$. **b** $n = 20$. **c** $n = 50$. **d** $n = 100$

the FPGA algorithm obtained a similar *purity*. However, observing the plot, the points produced by the NSMA algorithm seem to be near to those obtained by DMS and FPGA. We hence deduce that the latter algorithms produced only slightly better points.

Furthermore, in this problem NSMA outperformed the competitors in terms of Δ -*spread*. Indeed, our method managed to achieve a uniform Pareto front, as opposed to FPGA that produced most of the points in restricted areas of the objectives space.

As the value of n increases, the DMS performance gets worse and NSMA outperforms it w.r.t. all the metrics. In particular, in these cases our method turned out to be the best in terms of the *spread* metrics. For large values of n , DMS produced only a single point that is also dominated (it is not observable in the figure since it is too far from the reference front). In fact, the performance drop of DMS as the size of problems grows is not unexpected: derivative-free algorithms based on searches along coordinate directions are well known to poorly scale in general. The Δ -*spread* metric is not available for DMS in these cases, since it requires at least two points to be returned.

The performance of NSGA-II is rather poor, regardless the value of n . Arguably, this result can be attributed to the aforementioned NSGA-II performance slowdown occurring on problems characterized by a particularly large feasible sets (Sect. 3.1.1). Furthermore, as also commented in Sect. 4.2, in the MAN problem NSGA-II strug-

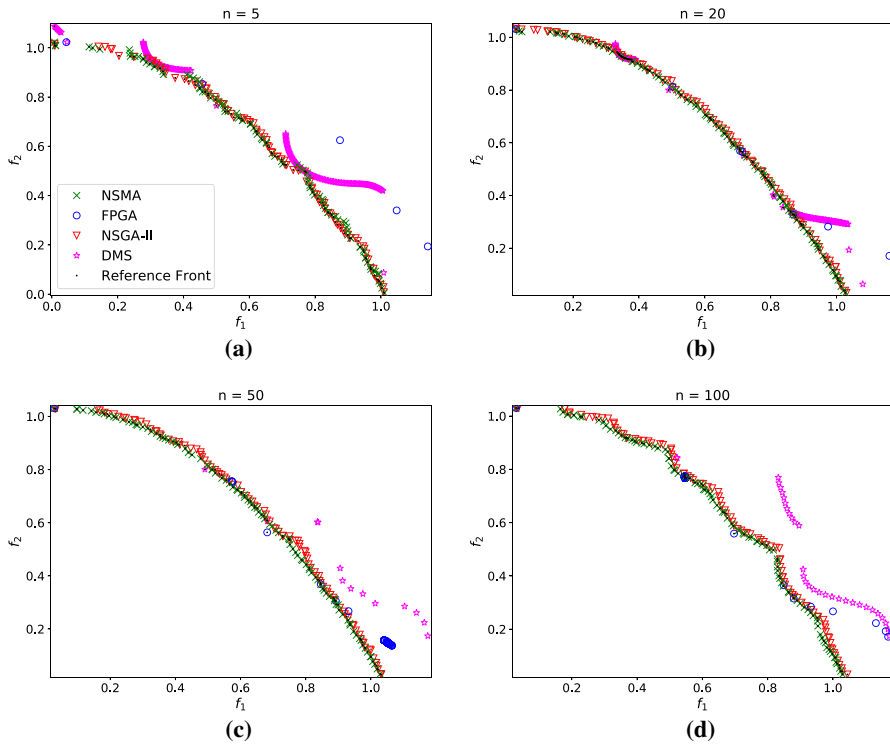


Fig. 6 Approximation of the Pareto front of the nonconvex CEC09_4 problem at different dimensionalities, retrieved by NSMA, FPGA, NSGA-II and DMS (for interpretation of the references to color in text, the reader is referred to the electronic version of the article). **a** $n = 5$. **b** $n = 20$. **c** $n = 50$. **d** $n = 100$

gles to explore the extreme regions of the objectives space. In this context, NSMA particularly exploited the surrogate bounds, the constrained steepest descent directions and the optimization of the points with high crowding distance. The constrained steepest descent directions also allowed FPGA to be the best algorithm in terms of *purity* overall. However, this method poorly performed regarding the *spread* metrics. Finally, for great values of n , our approach and FPGA are the only algorithms whose *purity* values are not equal to 0. Only NSMA managed to obtain points near to the ones of FPGA.

Regarding the CEC09_4 problem, whose results are reported in Figure 6 and Table 5, NSMA was the algorithm with the best overall performance: it generally obtained better metrics values than its most important competitors (FPGA and NSGA-II). Here, the combination of genetic operations and constrained steepest descent directions was greatly helpful to obtain remarkable results. Indeed, the independent use of only one of these two approaches did not lead to the same performance. In this problem, the DMS algorithm performed poorly regardless the value for n .

In conclusion, NSMA can be considered a viable option with convex problems, both in the low and the high dimensional cases. At the same time, our approach did not suffer with non-convex problems, as opposed to FPGA. On the contrary, it also

Table 5 Metrics values achieved by the four algorithms (NSMA, FFGA, NSGA-II and DMS) on the nonconvex CEC09_4 problem for $n = 5, 20, 50, 100$

n	METRIC	NSMA	FFGA	NSGA-II	DMS
5	<i>ND-points</i>	64	0	56	4
	<i>purity</i>	0.64	0.0	0.56	0.024
	Γ -spread	0.101	0.419	0.132	0.333
	Δ -spread	0.801	0.579	0.714	1.193
20	<i>ND-points</i>	86	4	49	30
	<i>purity</i>	0.86	0.571	0.49	0.234
	Γ -spread	0.086	0.475	0.037	0.3
	Δ -spread	0.546	0.68	0.537	1.613
50	<i>ND-points</i>	94	4	24	1
	<i>purity</i>	0.94	0.025	0.24	0.071
	Γ -spread	0.068	0.544	0.128	0.461
	Δ -spread	0.498	1.845	0.618	0.956
100	<i>ND-points</i>	96	109	13	0
	<i>purity</i>	0.96	0.948	0.13	0.0
	Γ -spread	0.136	0.514	0.143	0.491
	Δ -spread	0.641	1.757	0.701	1.248

The values marked in bold are the best obtained on a specific problem

outperformed NSGA-II, which is known to be a particularly suitable algorithm to use in these cases but struggles as the dimensionality of the problem grows.

4.5 Overall comparison

In this last section of computational experiments, we provide the performance profiles for the four considered algorithms on the entire benchmark of problems, listed in Table 1. The profiles are shown in Fig. 7.

The performance profiles remark once again the benefits of using our proposed approach. Regarding the *ND-points*, NSMA proved to be the most robust algorithm. This result was not obvious: we remind that our method, as opposed to FFGA and DMS, considers a fixed number of solutions in the population.

Another interesting result is related to the *purity* metric: NSMA is again the clear winner. In problems with complicated objective functions, local optimization of points in the NSMA mechanisms could result in a waste of computational time. From the results, however, we deduce that the converse is true: the combined use of constrained steepest descent directions and genetic operations allowed NSMA to achieve the best performance.

The proposed method also outperformed the other ones in terms of Γ -spread, while its performance is very similar to the one of NSGA-II in terms of Δ -spread. We can conclude that our approach is able to effectively obtain spread and uniform Pareto front approximations. At the same time, we deduce that the same cannot be said for the FFGA, which turned out to be the worst method w.r.t. the *spread* metrics. However,

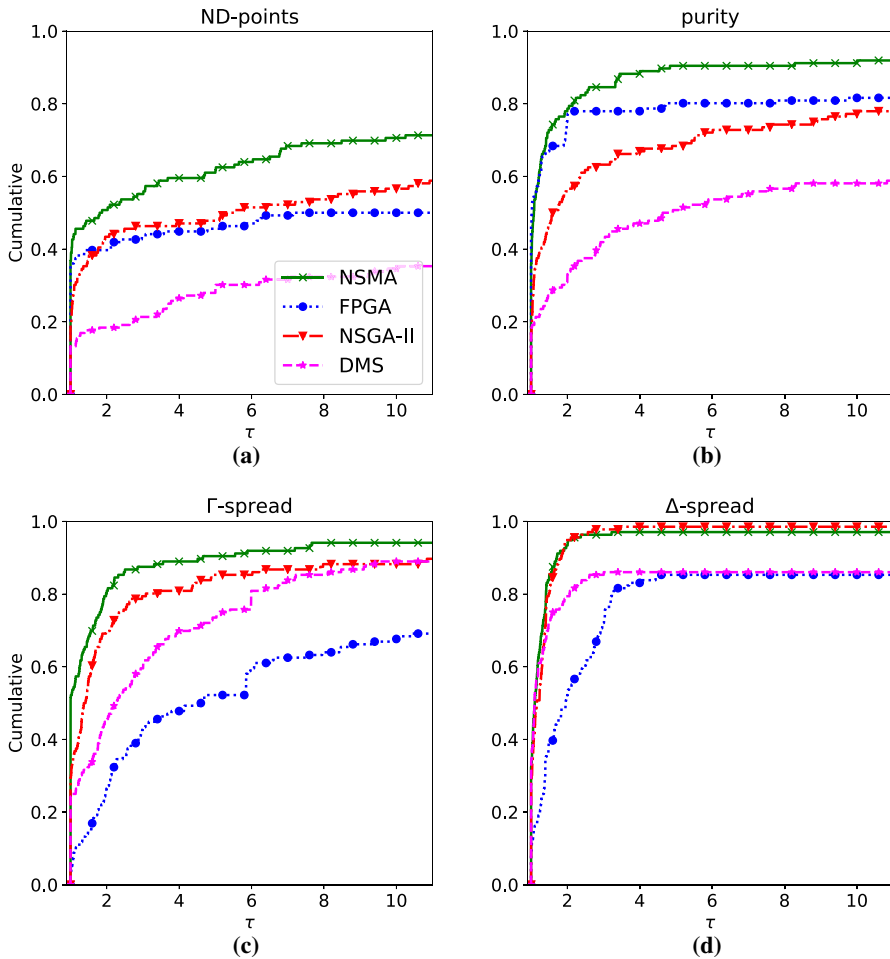


Fig. 7 Performance profiles for the NSMA, FPGA, NSGA-II and DMS algorithms on the CEC, ZDT, MOP and MAN problems, run with a time limit of 2 minutes (for interpretation of the references to color in text, the reader is referred to the electronic version of the article). **a** *ND-points*. **b** *purity*. **c** Γ -*spread*. **d** Δ -*spread*

the descent-based algorithm was the second best in terms of *purity*, outperforming NSGA-II. In general, DMS was not effective overall on the considered benchmark.

Lastly, we tested the four algorithms considering a time limit of 30 s for the experiments: the results can be seen in Fig. 8. Our aim is to observe the effectiveness of the methods at the first iterations.

Considering the *ND-points* and the *purity* metrics, we observe that the differences between our approach and the other algorithms are now even clearer, while the situation is not changed in terms of Γ -*spread*. Regarding the Δ -*spread* metric, NSGA-II was more effective than the other algorithms. However, our method was still competitive, as in terms of this metric it was the second most robust algorithm and it outperformed

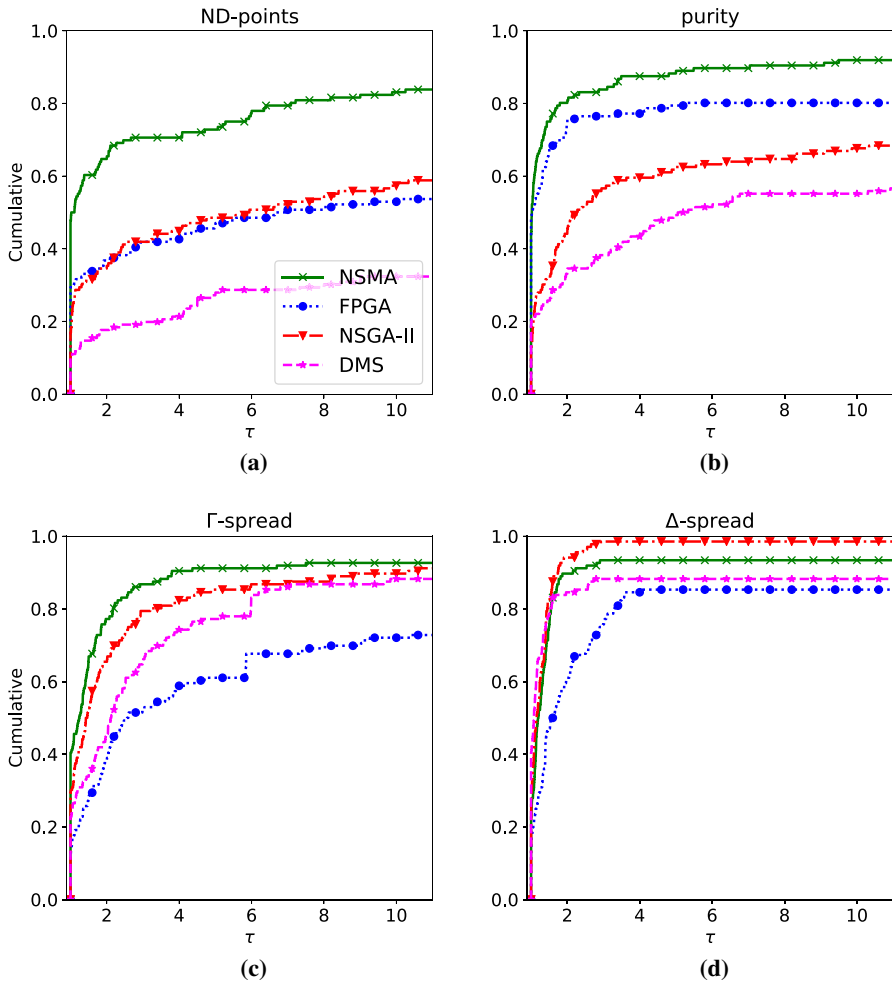


Fig. 8 Performance profiles for the NSMA, FPGA, NSGA-II and DMS algorithms on the CEC, ZDT, MOP and MAN problems, run with a time limit of 30 s (for interpretation of the references to color in text, the reader is referred to the electronic version of the article). **a** *ND-points*. **b** *purity*. **c** *Γ -spread*. **d** *Δ -spread*

FPGA and DMS. We can conclude that NSMA turned out to be also effective considering a smaller time limit: from the very first iterations, our approach was capable to obtain good, wide and uniform Pareto front approximations.

5 Conclusions

In this paper, we considered smooth multi-objective optimization problems subject to bound constraints. After a review of the existing literature, we listed and commented the main state-of-the-art approaches designed to approximate the Pareto front

of such problems, along with their benefits and drawbacks. In particular, we focused on NSGA-II [13], which is the most popular genetic algorithm, and on FPGA, which is a variant of the gradient-based descent method introduced in [10], capable of handling bound constraints. A detailed definition of FPGA, along with convergence properties, is provided in Appendix A. In a preliminary study, we compared these two algorithms trying to emphasize their strengths and weaknesses.

We then focused on the design of a memetic algorithm, whose aim is to combine the good features of both the aforementioned algorithms. We call this new method *Non-dominated Sorting Memetic Algorithm* (NSMA). In this procedure, we exploit the genetic operations of NSGA-II and the tools typical of gradient-based descent methods, such as the steepest descent directions and line searches. In particular, we employ a new descent method, called *Front Multi-Objective Projected Gradient* (FMOPG), which is a front-based variant of the original MOPG firstly introduced in [16]. For FMOPG, we proved properties of convergence to Pareto stationarity for the sequence of produced points.

Moreover, results of thorough computational experiments in which we compared our method with main state-of-the-art algorithms, including FPGA and NSGA-II, are provided. These results show that NSMA can consistently outperform its competitors in terms of popular metrics for multi-objective optimization. Our approach turned out to be highly effective in any considered setting.

Acknowledgements We are grateful to the editor and the anonymous referees for their precious comments that helped us to improve the quality of this manuscript. We would also like to thank Dr. Guido Cocchi, Dr. Giampaolo Liuzzi and Prof. Stefano Lucidi for the useful discussions.

Funding Open access funding provided by Università degli Studi di Firenze within the CRUI-CARE Agreement. The authors declare that no funds, grants, or other support were received during the preparation of this manuscript.

Data availability statement Data sharing not applicable to this article as no datasets were generated or analyzed during the current study.

Declarations

Conflict of interests The authors have no relevant financial or non-financial interests to disclose.

Code availability statement The full code of the experiments presented in this paper was made available for review. The implementation code of the NSMA algorithm proposed in this paper can be found at <https://github.com/pierlumanzu/nsma> (DOI: 10.5281/zenodo.7299857).

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

Appendix A: The Front Projected Gradient Algorithm

In this appendix, we describe the adaptation of the FSDA algorithm [10] to box-constrained optimization problems, which we call *Front Projected Gradient Algorithm* (FPGA). We initially report the scheme of the new adaptation. Then, in the remainder of the appendix, we provide a rigorous theoretical analysis.

A.1 Algorithmic scheme

We report the scheme of F PGA in Algorithm A1.

Algorithm A1: Front Projected Gradient Algorithm

```

1 Input:  $F : \mathbb{R}^n \rightarrow \mathbb{R}^m$ ,  $\Omega$  feasible closed and convex set,  $X_0$  set of feasible non-dominated points
   w.r.t.  $F$ .
2  $k = 0$ 
3 while a stopping criterion is not satisfied do
4    $\hat{X}_k = X_k$ 
5   for  $c = 1, \dots, |X_k|$  do
6     if  $x_c \in \hat{X}_k$  and  $\exists I \subseteq \{1, \dots, m\}$  such that
7       •  $x_c \in \hat{X}_k^I$ 
8       •  $\theta_\Omega^I(x_c) < 0$ 
9       then
10        Let  $d_{\Omega c}^I \in \mathcal{V}_\Omega^I(x_c)$  be the direction associated with  $\theta_\Omega^I(x_c)$ 
11         $\alpha = \text{B-FALS}(F(\cdot), \Omega, I, \hat{X}_k^I, x_c, d_{\Omega c}^I, \theta_\Omega^I(x_c))$ 
12         $\hat{X}_k = \hat{X}_k \setminus \{y \in \hat{X}_k \mid F(x_c + \alpha d_{\Omega c}^I) \not\preceq F(y)\} \cup \{x_c + \alpha d_{\Omega c}^I\}$ 
13      else
14        if  $x_c \in \hat{X}_k$  (i.e.  $x_c$  has not been filtered out by previous inner iterations) then
15           $x_c$  is a Pareto-stationary point w.r.t.  $F$  for Problem (1).
16     $X_{k+1} = \hat{X}_k$ 
17     $k = k + 1$ 
18 return  $X_k$ 

```

With respect to the FSDA algorithm, there are two major differences.

- For the descent directions, we solve instances of Problem (6), while in the original implementation Problem (3) is considered.
- We employ B-FALS (Algorithm 4) instead of FALS (Algorithm 2) in order to handle bound constraints.

In the remainder of the appendix, we provide some F PGA properties.

A.2 Algorithm analysis

In this subsection, we provide a formal analysis of the F PGA algorithm from a theoretical perspective.

We first prove the feasibility of the points produced by the algorithm.

Proposition A1 *Let $\{X_k\}$ be the sequence of sets of points generated by FPGA. Then, for all k , every point x_c in the set X_k is feasible for Problem (1).*

Proof The proof is straightforward. First of all, the initial set X_0 is composed by feasible points. New solutions are only added through Line 10. Considering that Ω is convex and $d_{\Omega_c}^I$ is a feasible direction by construction for any $I \subseteq \{1, \dots, m\}$, and reminding the stopping criteria of B-FALS, these new points are contained in Ω and, therefore, they are feasible for Problem (1). \square

In order to prove convergence properties, we need the concept of linked sequence (Definition 5) and the following assumption.

Assumption 1 Let X_0 be a set of feasible non-dominated points w.r.t. F . A point $x_0 \in X_0$ exists such that:

- x_0 is not Pareto-stationary w.r.t. F ;
- the set $\mathcal{L}(x_0) = \bigcup_{j=1}^m \{x \in \Omega : f_j(x) \leq f_j(x_0)\}$ is compact.

This latter one is similar to Assumption 1 in [10]. The difference is that, in this case, bound constraints must be also taken into account.

Proposition A2 *Let us assume that Assumption 1 holds. Let $\{X_k\}$ be the sequence of sets of non-dominated points w.r.t. F produced by FPGA. Let $\{x_k\}$ be a linked sequence, then it admits limit points and every limit point is Pareto-stationary w.r.t. F .*

Proof The proof is almost identical to the one of Proposition 5 in [10]. There is only one difference. After proving that

$$\lim_{\substack{k \rightarrow \infty \\ k \in K}} \alpha_{k+1} = 0 \quad (\text{A1})$$

[10, Equation 22], where K indicates a subsequence, the FSDA authors consider sufficiently large values of k such that $\alpha_{k+1} < \alpha_0$. In this case, the steps of FALS and the definition of X_k , $k \in K$, imply that there exists $y_k \in X_k$ such that

$$F_I(y_k) + \mathbf{1}\beta \frac{\alpha_{k+1}}{\delta} \theta^I(x_k) < F_I\left(x_k + \frac{\alpha_{k+1}}{\delta} d_k^I\right).$$

With respect to FALS, B-FALS has an additional stopping criterion: the step size must lead to a point that is feasible for Problem (1). In this context, $\alpha_{k+1}/\delta \leq \alpha_0$ might not have been selected because the point $x_k + (\alpha_{k+1}/\delta)d_{\Omega_k}^I \notin \Omega$. However, through a little modification, we can handle this additional stopping criterion.

First of all, Eq. (A1) still holds: the proof of this statement is the same provided in [10]. Then, we can consider sufficiently large values of k such that

$$\alpha_{k+1} < \frac{\alpha_{k+1}}{\delta} \leq 1.$$

In this way, since Ω is convex and $d_{\Omega^k}^I$ is a feasible direction by construction, the points produced by the two step sizes are feasible, i.e., the B-FALS feasibility stopping criterion is satisfied. Then, the steps of B-FALS and the definition of $X_k, k \in K$, imply that there exists $y_k \in X_k$ such that

$$F_I(y_k) + \mathbf{1}\beta \frac{\alpha_{k+1}}{\delta} \theta_{\Omega}^I(x_k) < F_I\left(x_k + \frac{\alpha_{k+1}}{\delta} d_{\Omega^k}^I\right).$$

From this point forward, we can follow the remainder of the proof of Proposition 5 in [10] in order to prove the thesis. \square

References

1. Bandyopadhyay, S., Saha, S., Maulik, U., Deb, K.: A simulated annealing-based multiobjective optimization algorithm: Aмоса. *IEEE Trans. Evol. Comput.* **12**(3), 269–283 (2008)
2. Bhuvana, J., Aravindan, C.: Memetic algorithm with preferential local search using adaptive weights for multi-objective optimization problems. *Soft Comput.* **20** (2015)
3. Brown, M., Smith, R.E.: Directed multi-objective optimization. *Int. J. Comput. Syst. Sign.* **6**(1), 3–17 (2005)
4. Cabassi, F., Locatelli, M.: Computational investigation of simple memetic approaches for continuous global optimization. *Comput. Oper. Res.* **72**, 50–70 (2016)
5. Campana, E.F., Diez, M., Liuzzi, G., Lucidi, S., Pellegrini, R., Piccialli, V., Rinaldi, F., Serani, A.: A multi-objective direct algorithm for ship hull optimization. *Comput. Optim. Appl.* **71**(1), 53–72 (2018)
6. Carrizo, G.A., Lotito, P.A., Maciel, M.C.: Trust region globalization strategy for the nonconvex unconstrained multiobjective optimization problem. *Math. Program.* **159**(1–2), 339–369 (2016)
7. Carrizosa, E., Frenk, J.B.G.: Dominating sets for convex functions with some applications. *J. Optim. Theory Appl.* **96**(2), 281–295 (1998)
8. Cocchi, G., Lapucci, M.: An augmented Lagrangian algorithm for multi-objective optimization. *Comput. Optim. Appl.* **77**(1), 29–56 (2020)
9. Cocchi, G., Lapucci, M., Mansueto, P.: Pareto front approximation through a multi-objective augmented Lagrangian method. *EURO J. Comput. Optim.* 100008 (2021)
10. Cocchi, G., Liuzzi, G., Lucidi, S., Sciandrone, M.: On the convergence of steepest descent methods for multiobjective optimization. *Comput. Optim. Appl.* 1–27 (2020)
11. Cocchi, G., Liuzzi, G., Papini, A., Sciandrone, M.: An implicit filtering algorithm for derivative-free multiobjective optimization with box constraints. *Comput. Optim. Appl.* **69**(2), 267–296 (2018)
12. Custódio, A.L., Madeira, J.A., Vaz, A.I.F., Vicente, L.N.: Direct multisearch for multiobjective optimization. *SIAM J. Optim.* **21**(3), 1109–1140 (2011)
13. Deb, K., Pratap, A., Agarwal, S., Meyarivan, T.: A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Trans. Evol. Comput.* **6**(2), 182–197 (2002)
14. Dolan, E.D., Moré, J.J.: Benchmarking optimization software with performance profiles. *Math. Program.* **91**(2), 201–213 (2002)
15. Drugan, M., Thierens, D.: Stochastic pareto local search: Pareto neighbourhood exploration and perturbation strategies. *J. Heuristics* **18** (2012)
16. Drummond, L.G., Iusem, A.N.: A projected gradient method for vector optimization problems. *Comput. Optim. Appl.* **28**(1), 5–29 (2004)
17. Drummond, L.G., Maculan, N., Svaiter, B.F.: On the choice of parameters for the weighting method in vector optimization. *Math. Program.* **111**(1–2), 201–216 (2008)
18. Eichfelder, G.: An adaptive scalarization method in multiobjective optimization. *SIAM J. Optim.* **19**(4), 1694–1718 (2009)
19. Filatovas, E., Lančinskas, A., Kurasova, O., Žilinskas, J.: A preference-based multi-objective evolutionary algorithm r-nsga-ii with stochastic local search. *CEJOR* **25**(4), 859–878 (2017)
20. Fliege, J., Drummond, L.G., Svaiter, B.F.: Newton’s method for multiobjective optimization. *SIAM J. Optim.* **20**(2), 602–626 (2009)

21. Fliege, J., Svaiter, B.F.: Steepest descent methods for multicriteria optimization. *Math. Methods Oper. Res.* **51**(3), 479–494 (2000)
22. Fliege, J., Vaz, A.I.F.: A method for constrained multiobjective optimization based on SQP techniques. *SIAM J. Optim.* **26**(4), 2091–2119 (2016)
23. Fukuda, E., Drummond, L.: A survey on multiobjective descent methods. *Pesquisa Operacional* **34**, 585–620 (2014)
24. Fukuda, E.H., Drummond, L.G.: On the convergence of the projected gradient method for vector optimization. *Optimization* **60**(8–9), 1009–1021 (2011)
25. Fukuda, E.H., Drummond, L.G.: Inexact projected gradient method for vector optimization. *Comput. Optim. Appl.* **54**(3), 473–493 (2013)
26. Fukuda, E.H., Drummond, L.G., Raupp, F.M.: A barrier-type method for multiobjective optimization. *Optimization*, 1–17 (2019)
27. Gonçalves, M.L.N., Lima, F.S., Prudente, L.F.: Globally convergent newton-type methods for multi-objective optimization. *Comput. Optim. Appl.* **83**(2), 403–434 (2022)
28. Gravel, M., Martel, J.M., Nadeau, R., Price, W., Tremblay, R.: A multicriterion view of optimal resource allocation in job-shop production. *Eur. J. Oper. Res.* **61**(1–2), 230–244 (1992)
29. Gribel, D., Vidal, T.: Hg-means: A scalable hybrid genetic algorithm for minimum sum-of-squares clustering. *Pattern Recogn.* **88**, 569–583 (2019)
30. Grosso, A., Locatelli, M., Schoen, F.: A population-based approach for hard global optimization problems based on dissimilarity measures. *Math. Program.* **110**(2), 373–404 (2007)
31. Hu, X., Huang, Z., Wang, Z.: Hybridization of the multi-objective evolutionary algorithms and the gradient-based algorithms. In: *The 2003 Congress on Evolutionary Computation*, 2003. CEC'03., vol. 2, pp. 870–877. IEEE (2003)
32. Huband, S., Hingston, P., Barone, L., While, L.: A review of multiobjective test problems and a scalable test problem toolkit. *IEEE Trans. Evol. Comput.* **10**(5), 477–506 (2006)
33. Kim, H., Liou, M.S.: Adaptive directional local search strategy for hybrid evolutionary multiobjective optimization. *Appl. Soft Comput. J.* **19**, 290–311 (2014)
34. Lara, A., Sanchez, G., Coello, C.A.C., Schutze, O.: Hcs: A new local search strategy for memetic multiobjective evolutionary algorithms. *IEEE Trans. Evol. Comput.* **14**(1), 112–132 (2010)
35. Laumanns, M., Thiele, L., Deb, K., Zitzler, E.: Combining convergence and diversity in evolutionary multiobjective optimization. *Evol. Comput.* **10**(3), 263–282 (2002)
36. Liu, D., Tan, K.C., Goh, C.K., Ho, W.K.: A multiobjective memetic algorithm based on particle swarm optimization. *IEEE Trans. Syst. Man Cyberne. B (Cybern.)* **37**(1), 42–50 (2007)
37. Liu, T., Gao, X., Yuan, Q.: An improved gradient-based nsga-ii algorithm by a new chaotic map model. *Soft. Comput.* **21**(23), 7235–7249 (2017)
38. Liuzzi, G., Lucidi, S., Rinaldi, F.: A derivative-free approach to constrained multiobjective nonsmooth optimization. *SIAM J. Optim.* **26**(4), 2744–2774 (2016)
39. Locatelli, M., Maischberger, M., Schoen, F.: Differential evolution methods based on local searches. *Comput. Oper. Res.* **43**, 169–180 (2014)
40. Locatelli, M., Schoen, F.: *Global optimization: theory, algorithms, and applications*. SIAM (2013)
41. Mandal, S.K., Pacciarelli, D., Løkketangen, A., Hasle, G.: A memetic NSGA-II for the bi-objective mixed capacitated general routing problem. *J. Heuristics* **21**(3), 359–390 (2015). Number: 3
42. Mansueto, P.: NSMA: A memetic procedure for global multi-objective optimization (2022). <https://doi.org/10.5281/zenodo.7299857>
43. Mansueto, P., Schoen, F.: Memetic differential evolution methods for clustering problems. *Pattern Recogn.* **114**, 107849 (2021)
44. Mostaghim, S., Branke, J., Schmeck, H.: Multi-objective particle swarm optimization on computer grids. In: *Proceedings of the 9th Annual Conference on Genetic and Evolutionary Computation*, pp. 869–875. ACM (2007)
45. Palermo, G., Silvano, C., Valsecchi, S., Zaccaria, V.: A system-level methodology for fast multi-objective design space exploration. In: *Proceedings of the 13th ACM Great Lakes Symposium on VLSI*, pp. 92–95. ACM (2003)
46. Pascoletti, A., Serafini, P.: Scalarizing vector optimization problems. *J. Optim. Theory Appl.* **42**(4), 499–524 (1984)
47. Pellegrini, R., Campana, E., Diez, M., Serani, A., Rinaldi, F., Fasano, G., Iemma, U., Liuzzi, G., Lucidi, S., Stern, F.: Application of derivative-free multi-objective algorithms to reliability-based robust

- design optimization of a high-speed catamaran in real ocean environment I. *Engineering Optimization* IV-Rodrigues et al.(Eds.) p. 15 (2014)
48. Shukla, P.K.: On gradient based local search methods in unconstrained evolutionary multi-objective optimization. In: *International Conference on Evolutionary Multi-Criterion Optimization*, pp. 96–110. Springer (2007)
 49. Sindhya, K., Miettinen, K., Deb, K.: A hybrid framework for evolutionary multi-objective optimization. *IEEE Trans. Evol. Comput.* **17**(4), 495–511 (2013)
 50. Sun, Y., Ng, D.W.K., Zhu, J., Schober, R.: Multi-objective optimization for robust power efficient and secure full-duplex wireless communication systems. *IEEE Trans. Wireless Commun.* **15**(8), 5511–5526 (2016)
 51. Tavana, M.: A subjective assessment of alternative mission architectures for the human exploration of Mars at NASA using multicriteria decision making. *Comput. Oper. Res.* **31**(7), 1147–1164 (2004)
 52. Tiwari, S., Fadel, G., Koch, P., Deb, K.: Performance assessment of the hybrid archive-based micro genetic algorithm (amga) on the cec09 test problems. In: *2009 IEEE Congress on Evolutionary Computation*, pp. 1935–1942 (2009)
 53. Villalobos-Cid, M., Dorn, M., Ligabue-Braun, R., Inostroza-Ponta, M.: A memetic algorithm based on an nsga-ii scheme for phylogenetic tree inference. *IEEE Trans. Evol. Comput.* **23**(5), 776–787 (2018)
 54. Wang, X., Hirsch, C., Kang, S., Lacor, C.: Multi-objective optimization of turbomachinery using improved nsga-ii and approximation model. *Comput. Methods Appl. Mech. Eng.* **200**(9–12), 883–895 (2011)
 55. White, D.: Epsilon-dominating solutions in mean-variance portfolio analysis. *Eur. J. Oper. Res.* **105**(3), 457–466 (1998)
 56. Zhang, Q., Zhou, A., Zhao, S., Suganthan, P., Liu, W., Tiwari, S.: Multiobjective optimization test instances for the cec 2009 special session and competition. *Mech. Eng.* (2008)
 57. Zitzler, E., Deb, K., Thiele, L.: Comparison of multiobjective evolutionary algorithms: empirical results. *Evol. Comput.* **8**(2), 173–195 (2000)

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.